VYSOKÁ ŠKOLA BÁŇSKÁ – TECHNICKÁ UNIVERZITA OSTRAVA

EKONOMICKÁ FAKULTA

KATEDRA APLIKOVANÉ INFORMATIKY

Systém centrální správy výsledkového servisu sportovních událostí

Central System of Sporting Events Results Services

Student: Jiří Fluksa

Vedoucí diplomové práce: RNDr. Ivo Martiník, Ph.D.

Ostrava 2013

VŠB - Technická univerzita Ostrava
Ekonomická fakulta
Katedra aplikované informatiky

Zadání diplomové práce

Student: **Bc. Jiří Fluksa**

Studijní program: N6209 Systémové inženýrství a informatika

Studijní obor: 1802T001 Aplikovaná informatika

Téma: Systém centrální správy výsledkového servisu sportovních událostí
Central System of Sporting Events Results Services

Zásady pro vypracování:

1. Úvod
2. Metodická východiska a nástroje
3. Analýza současného stavu výsledkového servisu
4. Návrh a realizace systému
5. Závěr

Seznam použité literatury
Seznam zkratek
Prohlášení o využití výsledků diplomové práce
Seznam příloh
Přílohy

Seznam doporučené odborné literatury:

FOWLER, Martin. *Destilované UML*. Praha: Grada, 2009. ISBN 978-80-247-2062-3.
FREEMAN, Adam. *Pro ASP.NET MVC 4*. 4th ed. Berkley: Apress, 2012. ISBN 978-1-4302-4236-9.
GALLOWAY, Jon et al. *Professional ASP.NET MVC 4*. Indianapolis: Wiley, 2012. ISBN 978-1-118-42432-2.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **RNDr. Ivo Martiník, Ph.D.**

Datum zadání: 22.11.2013
Datum odevzdání: 25.04.2014

VYSOKÁ ŠKOLA BÁŇSKÁ-TECHNICKÁ UNIVERZITA OSTRAVA Ekonomická fakulta 1

Ing. Petr Rozehnal, Ph.D.
vedoucí katedry

prof. Dr. Ing. Dana Dluhošová
děkanka fakulty

„Prohlašuji, že jsem celou práci včetně všech příloh vypracoval samostatně."

V Ostravě dne 25.4.2014 ..... Fluksa Jiří

Bc. Jiří Fluksa

Poděkování:

Děkuji panu RNDr. Ivo Martiníkovi, Ph.D. a Ing. Petru Blažejovi za odborné vedení práce a jejich cenné rady a připomínky.

Obsah

1 Úvod

Dlouho si při měření závodů pořadatelé vystačili pouze s ručními stopkami, tužkou a papírem. I když se u některých menších okresních závodů používají stále, obyčejné stopky přestaly být přesné, když při vzrůstajícím počtu účastníků závodů (na největším českém běhu – Volkswagen Maraton Praha - se počet účastníků blíží hranici 10 000) přišel požadavek na zachycení takzvaných reálných časů. Ty uvádějí výkon závodníka od průběhu startovní čárou po protnutí čáry cílové.

Současnému měření časů při velkých městských bězích (a nejen u nich) dominují moderní technologie. Využívá se takzvané čipové časomíry, kdy závodníci mají čip umístěný na noze na textilní pásce, nebo jej mají připnutý na startovním čísle a probíhají přes měřící místa, které zajišťují UHF antény (zpravidla umístěné na trati v tenkém gumovém obalu), tyto snímají jejich čipy a RFID přijímače za pomocí speciálního softwaru určují závodníkův čas, jeho průměrnou rychlost, uběhnuté okruhy, mezičasy, případně další potřebné veličiny, které je možno vypočítat z naměřených dat.

Současně s příchodem moderních technologií vzešel i požadavek na jednoduché, rychlé a přehledné zobrazení výsledků těchto závodů. Účastníci těchto závodů, které jsou měřené prostřednictvím čipové časomíroy, očekávají, že firma která provádí měření závodu zajistí již v průběhu závodu průběžné výsledky, případně v krátké době po skončení závodu, zveřejní naměřené výsledky.

Cile práce:

1. Studie a popis technologií ASP .NET, ASP .NET MVC a dalších souvisejících technologií,
2. analýza návrhu webové aplikace pro zobrazování výsledků měřených sportovních událostí firmou On line System,
3. provedení návrhu a realizace aplikace.

Cílem této práce je tedy vytvoření webové aplikace pro prezentaci výsledků sportovních událostí měřených firmou On line System, provedení analýzy a nárhu budoucího systému, jeho napojení na aktuální databázi akcí pro systém správy akcí, provedení realizace systému, jeho implementace a spuštění na firemním serveru.

První část práce se zaměřuje na základní popis pojmů spojených s webovými aplikacemi a internetem obecně. Rovněž popisuje základní technologie, nástroje a postupy, které byly využity při tvorbě této webové aplikace.

V druhé části je provedena analýza současného stavu výsledkového servisu. Jsou zde rovněž uvedeny konkrétní požadavky na novou webovou aplikaci a analytický návrh budoucí webové aplikace.

Poslední část práce popisuje již konkrétní kroky budování, implementace a provozu výsledné webové aplikace na firemním serveru. Rovněž popisuje základní logiku a strukturu aplikace a seznámí nás také se vzhledem a rozvržením, při jehož tvorbě byla zahrnuta i podpora mobilních zařízení, která v poslední době zažívají obrovský rozvoj a těší se stále větší a větší základně uživatelů.

2 Metodická východiska a nástroje

Následující část popisuje základní definice, nástroje a pojmy, které lze v kontextu této práce pokládat za důležité. Jednak proto, že se týkají dané problematiky a jednak z důvodu pochopení významu pojmů, se kterými se v této diplomové práci pracuje.

2.1 Základní pojmy

2.1.1 Internet

Počátkem šedesátých let v době vrcholící studené války se začaly ve Spojených státech amerických objevovat myšlenky na vytvoření sítě, která by navzájem propojovala nejdůležitější vojenské, vládní a vědecko-výzkumné počítače. Motivace byla ve skrze vojenská – přežití v případě jaderného úderu. Z toho vyplynul i základní požadavek, aby šlo o síť decentralizovanou, která bude schopna fungovat i v případě výpadku některého z uzlů sítě. [6]

Všechny zprávy měly být posílány po kouscích (paketech), každý paket měl být vybaven adresou určení. Pakety měly být posílány mezi uzly sítě. Na těchto principech v roce 1969 vznikla spojením 4 amerických univerzit první síť – ARPANET.

Důležitými mezníky v dalším vývoji se stávají roky 1983, kdy ARPANET přechází na protokol TCP/IP a 1984 zavedením dodnes platného doménového systému DNS (Domain Name Services). [6]

Po roce 1983 došlo k nastartování růstu počtu zapojených počítačů. V roce 1984 byla překročena hranice 1 000 uzlů, počet 10 000 padl v roce 1987 a o dva roky později již 100 000. Hranice 1 000 000 pak byla překročena v roce 1992. I přes tyto nebývalá tempa růstu až do roku 1993 zůstával Internet doménou vědeckých a akademických pracovišť. Situace se začala měnit v roce 1991, kdy americký Kongres přijal zákon *High Performance Computing Act.* Od roku 1993 se na Internetu začaly ve velkém objevovat komerční firmy, nejprve počítačové a později i z dalších oborů. [6]

Současnou podobu Internetu lze definovat jako globální informační systém, který:

- je logicky propojen do jednoho celku prostřednictvím globálního adresného prostoru založeného na protokolu IP (Internet Protocol) nebo jeho následných rozšířeních,

- je schopen podporovat komunikaci prostřednictvím rodiny protokolů TCP/IP (Transmission Control Protocol/Internet Protocol) nebo jeho následných rozšíření nebo jiných protokolů kompatibilních s rodinou protokolů TCP/IP,
- nabízí veřejné nebo privátní dostupné služby vyšší úrovně, které jsou založeny na komunikační a další infrastruktuře.

Internet prošel během své existence mnoha proměnami. Vznikal ještě v době sdílení času procesorů, ale nyní se musí vypořádávat s érou osobních počítačů, architektury klient/server, komunikací typu peer-to-peer nebo síťovými stanicemi. Byl zamýšlen jako podpůrný prostředek od sdílení souborů a vzdáleného přihlašování až ke sdílení souborů a spolupráci, přinesl elektronickou poštu a World Wide Web. [6]

2.1.2 World Wide Web

World Wide Web je služba založená na architektuře klient/server, která v prostředí Internetu zpřístupňuje hypertextové dokumenty. Služba, jejímž autorem je Tim Berners-Lee, se zrodila v roce 1990 v rámci organizace CERN, kde vznikl požadavek na snadno ovladatelný a všem přístupný vývěskový systém. Systém navržený původně výhradně pro potřeby fyziků se za několik málo let vypracoval v nejpoužívanější službu internetu. [6]

Až do roku 1993 se služba WWW využívala jen v malé míře. Software pro chod služby vznikal především v akademickém prostředí, a tím byla dána i jejich strohost z hlediska uživatelské přívětivosti. Přelom nastává na konci roku 1993, kdy se objevil první grafický prohlížeč NCSA Mosaic. V prosinci 1994 byl uveden na trh první Netscape Navigator. Pro domácí použití a ve vzdělávací sféře byl nabízen zdarma, což lze považovat za zlomový okamžik pro masivní rozšíření Internetu mimo akademickou sféru do firem a domácností. [6]

2.1.1 Webová aplikace

V literatuře, na internetu a dalších zdrojích lze nalézt velké množtví definic webové aplikace. Pro představu jsou zde dvě uvedeny.

Webová aplikace je HTML nebo XHTML stránka využívající kód spuštěný na serveru, který pohotově reaguje na požadavky klienta a dynamicky tvoří stránky HTML zobrazující se v okně prohlížeče spuštěného na klientském počítači. K implementaci je využíváno různých nástrojů (PHP, ASP.NET apod.), které umožňují realizovat běžnou funkcionalitu webových stránek (internetový obchod, počítadlo návštěv, ankety atd.). [4]

„Webová aplikace (Web Application) je taková aplikace, kterou není nutno instalovat na zařízení uživatele (počítač, tablet, smartphone) a můžete ji spustit z kteréhokoliv zařízení pomocí webového prohlížeče, protože je spuštěna na straně serveru. Vzhledem k tomu, že je třeba jen prohlížeč se webová aplikace někdy nazývá též jako lehký klient.“ [11]

Výhody webových aplikací:

- nemusí se instalovat,
- uživatel nemusí nic aktualizovat (aktualizace probíhá na serveru),
- potřebujete pouze webový prohlížeč,
- data jsou uchovávána a zálohována na serveru a jsou přístupná odkudkoliv,
- můžete k nim přistupovat odkudkoliv i z cizího počítače.

Nevýhody webových aplikací:

- vyžadují připojení na internet,
- někdy pomalejší tok dat a práce s aplikací (rychlost je závislá na kvalitě připojení),
- možná bezpečnostní rizika úniku dat v případě nekvalitního poskytovatele.

2.2 ASP .NET

Webové aplikace je možné vytvářet za pomocí různých dostupných technologií. Jelikož firma, pro kterou je tato aplikace vytvářena, využívá technologii ASP .NET, stala se tato nejvhodnější variantou.

ASP .NET, nástupce ASP (Active Server Pages), byl poprvé vydán v roce 2002. Nejednalo se však o vylepšení původního ASP, ale o zcela novou technologii, díky jejímu začlenění v ASP .NET Frameworku. [5]

.NET Framework je základní komponentou .NET (zastřešující název pro soubor technologií v softwarových produktech společnosti Microsoft, které tvoří celou platformu). Jde o prostředí potřebné pro běh aplikací a nabízející jak spouštěcí rozhraní, tak potřebné knihovny. Pro vývoj .NET aplikací vydal Microsoft Visual Studio .NET. Poprvé byl .NET

Framework představen v únoru 2002. Aktuální verze, vydaná v říjnu 2013, nese označení verze 4.5.1.[1]

2.2.1 7 základních vlastností ASP .NET

ASP .Net pracuje jiným způsobem než tradiční skriptovací technologie jako ASP nebo PHP. Mezi základní odlišnosti patří:

- **ASP .NET je integrován s .NET Frameworkem** – .NET Framework je rozčleněn do kolekce funkčních částí zahrnující tisíce typů. Třídy jsou seskupeny do logických hierarchických skupin, kterým se říká jmenný prostor (namespace).
- **ASP .NET se neinterpretuje, ale kompiluje –** Jedním z hlavních důvodů degradace výkonu ve skriptech ASP je to, že se v kódu webových stránek ASP používají interpretované skriptovací jazyky. To znamená, že když se aplikace vykonává, musí skriptovací hostitel na serveru interpretovat kód a přeložit jej do strojového kódu nižší úrovně. Tento proces je ovšem velmi pomalý.

  Aplikace ASP .NET prochází svěma kompilačními etapami. V první etapě se kód zkompiluje do přechodného jazyka, který se nazývá Intermediate Language (IL). Druhá úroveň kompilace nastává těsně předtím, než se stránka skutečně vykoná. V tomto okamžiku se kód IL zkompiluje do nativního nízkoúrovnňového strojového kódu. Tato etapa se nazývá jako kompilace just-in-time (JIT).
- **ASP .NET podporuje práci ve více programovacích jazycích –** aplikace ASP .NET je možné psát v kterémkoli z podporovaných jazyků .NET (např. C#, J#, Visual Basic). Toto je umožněno díky překladu kteréhokoli použitého jazyka do IL.
- **ASP .NET běží uvnitř společného runtime programovacích jazyků –** nejdůležitějším aspektem ASP .NET je, že běží uvnitř runtimového prostředí CLR. Z tohoto plyne několik výhod:
- **Automatická správa paměti a svoz odpadků (garbage collection) –** Vždy, když aplikace vytvoří instanci nějaké třídy, CLR alokuje pro objekt prostor na řízené haldě (managed heap). Uvnitř CLR pravidelně běží „garbage collector“ a automaticky uvolňuje nepoužívanou paměť objektů, k nimž už nelze přistupovat (reference na objekt se dostane mimo obor nebo aplikace končí).

[1] .NET Framework Versions and Dependencies:
http://msdn.microsoft.com/en-us/library/bb822049%28v=vs.110%29.aspx

- **Typová bezpečnost –** Když kompilujeme nějakou aplikaci, přidá .NET do assembly (zkompilovaná knihovna tříd používána pro nasazení, správu verzí a bezpečnost) informaci specifikující některé podrobnosti o ní. Například dostupné třídy, jejich členy, jejich datové typy atd. výsledkem je, že assembly zkompilovaného kódu budou plně soběstačné. Tato vrstva také zamezuje vzniku různých nízkoúrovňových chyb.
- **Rozšiřitelná metadata –** Informace o třídách a členech jsou pouze jedním z typů metadat, která .NET ukládá do zkompilované assembly. Metadata popisují kód a umožňují poskytnout dodatečné informace pro runtime nebo pro jiné služby. Metadata mohou například sdělit debuggeru, jak má trasovat kód, nebo mohou sdělit Visual Studiu, jak má zobrazit nějaký prvek v režimu designu.
- **Strukturované zpracování chyb –** Pomocí strukturovaného zpracování výjimek je možné kód, jímž se bude reagovat na chyby, uspořádat logicky, stručně a výstižně. Je možné vytvářet oddělené bloky, v nichž se budou zpracovávat různé druhy chyb. Ovladače výjimek je možné vnořovat do hloubky, přes více vrstev.
- **Multithreading –** CLR poskytuje fond vláken, který mohou využívat různé třídy. Je možné volat metody, číst soubory, nebo komunikovat s webovými službami asynchronně, bez potřeby explicitního vytváření nového vlákna.
- **ASP .NET je objektově orientované –** mimo to, že kód ASP .NET má úplný přístup ke všem objektům v .NET Frameworku, můžeme rovněž těžit z veškerých konvencí objektově orientovaného prostředí (OOP). Lze například vytvářet opětovně využitelné třídy, standardizovat kód rozhraními, rozšiřovat existující třídy děděním, nebo začlenit nějakou užitečnou funkcionalitu do zkompilované komponenty určené k dalšímu šíření.
- **ASP .NET podporuje různá zařízení a různé prohlížeče –** velkou výzvou pro webové vývojáře je značná různorodost prohlížečů, které je potřeba podporovat. Různé prohlížeče, jejich verze a jejich konfigurace podporují standard HTML různě. ASP.NET se vypořádává s tímto problémem velmi inteligentně. Disponuje bohatě vybavenou skupinu webových ovládacích prvků. Ty realizují svůj kód HTML adaptivně, což znamená, že berou v úvahu schopnosti klienta.
- **ASP .NET se snadno rozmisťuje a konfiguruje –** Každá instalace .NET Frameworku poskytuje stejné jádro tříd. Z toho plyne, že aplikace ASP.NET se rozmisťují relativně snadno. Ve většině případů stačí pouze zkopírovat všechny soubory do nějakého virtuálního adresáře

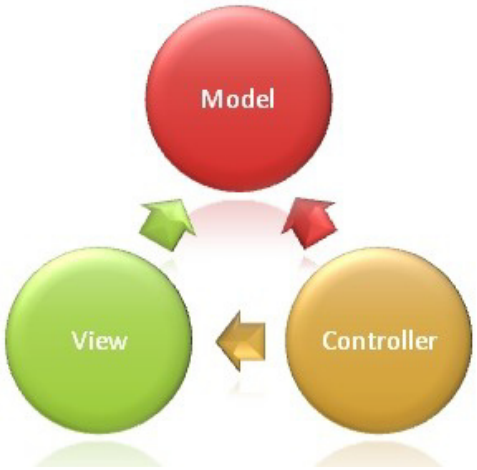
na ostrém serveru. Pokud má hostitelský stroj .NET Framework, nejsou potřeba žádné registrační kroky, které by mohly být časově velmi náročné.

ASP .NET fázi konfigurace usnadňuje tím, že minimalizuje závislost na nastavení v IIS (Internet Information Services). Většina nastavení ASP .NET se ukládá do souboru web.config, který existuje právě pro tento účel. Soubor web.config se umísťuje do téhož adresáře, kde jsou umístěny webové stránky. Obsahuje hierarchická seskupení nastavení aplikace, která jsou uložená ve formátu XML. [5]

2.3 ASP .NET MVC

Model-View-Controller (MVC) je návrhový vzor architektury, který rozděluje aplikační logiku do 3 prvků (viz obrázek 2.1)

Obr. 2.1 Schéma Model – View – Controller



Zdroj: http://www.w3schools.com/aspnet/mvc_intro.asp

- **Model** je objekt reprezentující datovou strukturu, se kterou aplikace pracuje.
- **View** je zodpovědné za poskytování uživatelského rozhraní uživateli aplikace. Je dáno odkazem na model a tento transformuje do formátu vhodného pro uživatele dle jeho potřeb. Veškerou práci nad daty provádí Controller a úkolem View je tedy pouze jejich prezentace uživateli. View by nemělo obsahovat ani aplikační, ani doménovou logiku. Zároveň by mělo obsahovat co nejméně kódu, ale je zde možno implementovat prezentační logiku.
- **Controller** je nejdůležitější a nejsložitější částí v ASP .NET MVC, která řídí celý běh aplikace. Obsahuje aplikační logiku v souvislosti s akcemi, které mohou být vyvolány uživatelem z View. Controller rovněž určuje, který View bude použit k výpisu dat nebo

interakci s uživatelem. Podle konvencí aplikačního rámce ASP .NET MVC by měla existovat třída Controlleru pro každou podsložku obsahující View. Název třídy takovéhoto Controlleru se značí jako název dané podsložky s přídavkem Controller. Pokud tedy bude stránka zobrazující výsledky závodu s názvem *Vysledky.cshtml* ve složce *Vysledky*, k ní bude náležet třída označená jako *VysledkyController.cs*, která bude obsahovat veřejnou metodu *Vysledky*. [2] [3]

ASP .NET MVC je framework pro vývoj webových aplikací na platformě Microsoft .NET, který představuje alternativu k Web Forms a poskytuje vývojářům způsob vytvářet dobře strukturované webové aplikace. Poprvé byl vydán 13. Května 2009. Druhá verze byla vydána o rok později v březnu 2010 a přinesla několik klíčových funkcí. Předně to byli pomocníci generování uživatelského rozhraní s nastavitelnými šablonami, validace atributů modelu na serverové i klientské straně a silně typové HTML pomocníky. Dále tato verze rovněž přinesla mnoho nových vlastností aplikačního rozhraní, jako podpora pro rozdělení velkých aplikací do oblastí, podpora asynchronních controllerů, podpora renderování podsekce na stránce pomocí Html.RenderAction a mnoho dalších pomocných funkcí, nástrojů a vylepšení API. O necelý rok později, v lednu 2011, byla vydána již třetí verze ASP .NET MVC, která přinesla vylepšení validace modelu, lepší podporu JavaScriptu, JSON bindování, podporu pro .NET 4 data anotace a Razor view engine (viz 2.4.1). [2] [3]

2.4 ASP .NET MVC 4

MVC 4 byl vydán v srpnu 2012 a jeho novinky se zaměřují na některé vyspělejší scénáře. Přináší nový framework Web API pro tvorbu HTTP služeb, který může využít široká škála klientských systémů, včetně prohlížečů a mobilních zařízení. Dalším zkvalitněním je vylepšení výchozí šablony projektu, která kromě modernějšího vzhledu dostala rovněž vylepšenou funkcionalitu. Šablona nyní využívá tzv. „reagující vzhled“ (Responsive Design), který se svým rozvržením přizpůsobuje desktopovým a mobilním prohlížečům.

Kromě „reagujícího vzhledu“ nabízí MVC 4 rovněž novinku v podobě šablony projektu pro mobilní zařízení. Tato šablona je založena na technologii jQuery Mobile a open-source knihovně pro tvorbu aplikací optimalizovaných pro dotykové ovládání. Pro správné zobrazení stránky na různých zařízeních je rovněž možné využít rozdílné režimy zobrazení. Aplikace si v tomto případě vybere View na základě prohlížeče, který podává žádost. Pokud tedy například aplikace zaznamená požadavek desktopového prohlížeče, použije

`Views\Home\Index.cshtml` šablonu. V případě mobilního prohlížeče může aplikace vrátit `Views\Home\Index.mobile.cshtml` šablonu.

Dalšími novinkami jsou sdružování (Bundling) a optimalizace kódu (Minification). Sdružováním je možné spojit více souborů do jednoho souboru. Sdružování zlepší výkon webového požadavku, protože bude obsahovat méně souborů, tedy se zmenší počet HTTP požadavků. Optimalizace kódu pro skripty a CSS soubory odstraňuje nepotřebné bílé místa, komentáře a zkracuje názvy proměnných do jednoho znaku. Tyto novinky je možné povolit nebo zakázat nastavením hodnoty kompilace na `true` nebo `false` ve `web.config` souboru. [2] [10]

2.4.1 Razor View Engine

Razor View Engine byl představen s ASP .NET MVC3 a jde o výchozí zobrazení pohledů. Razor se stal odpovědí na jeden z nejvíce požadovaných návrhů a sice poskytnout čisté, lehké, a jednoduché zobrazení, které neobsahuje zbytečné syntaktické prvky.

Klíčovým přechodným znakem v razoru je „zavináč“ (@). Tento znak se používám k přechodu od Html značky ke kódu. Existují dva základní typy přechodů: výrazy kódu a bloky kódu.

Přípona .cshtml značí, že pohled bude zpracovávat Razor. Předchozí verze MVC spoléhaly na zobrazení enginu ASPX, jehož pohledy mají příponu .aspx. [3]

Razor `@(Html.Metoda<Typ>())`

Web Forms `<%: Html.Metoda<Typ>() %>`

2.4.2 Zpracování adres

Směrovací systém pracuje pomocí sady tras. Aplikace si při startu vytvoří tzv. routing table, která obsahuje sadu šablon, dle kterých se řídí při zpracování požadavků. Tyto šablony jsou definovány v metodě `MapRoute()`, která je volána během startu aplikace.

Následující ukázka nám představuje výchozí šablonu, vytvořenou při založení projektu:

```
routes.MapRoute(
          name: "Default", // Název cesty
          url: "{controller}/{action}/{id}", // šablona URL
          defaults: new { controller = "Home", action = "Index",
            id = UrlParameter.Optional } // výchozí hodnoty
                 );
```

Tuto šablonu je možné měnit, případně nahradit vlastní, pokud je potřeba. V tomto případě je nutné dodržet správné pořadí a definovat šablony od nejvíce specifické až po nejvíce obecnou, z důvodu vyhodnocování šablon v pořadí v jakém byly přidány. [2] [3]

2.5 Další technologie spolupracující s ASP .NET MVC

2.5.1 AJAX

ASP .NET MVC4 je moderní webový framework, který podporuje technologii Ajax (Asynchronous JavaScript and XML). Ajax je obecné označení pro technologie vývoje interaktivních webových aplikací, které mění svůj obsah stránek bez nutnosti jejich kompletního znovunačítání za pomocí asynchronního zpracování webových stránek. Jádro podpory Ajaxu pochází z open source jQuery JavaScriptové knihovny.

Mezi výhodu AJAXu patří odstranění nutnosti znovunačítání a překreslení celé stránky při každé operaci, které jsou nutné u klasického modelu WWW stránek. Pokud například uživatel provede filtrování tabulky, celá stránka se musí znovu načíst ze serveru, třebaže se na ní jen aktualizuje tabulka a zbytek obsahu zůstává stejný. Prostřednictvím AJAXu proběhne filtrování na pozadí, server zašle jen ty části stránky, které se změnily, a jen tyto části se uživateli na stránce aktualizují. S AJAXem má uživatel pocit větší plynulosti práce. Z tohoto rovněž vyplývá potenciál snížení zátěže webových serverů a sítě obecně, jelikož není nutné při každém požadavku sestavit celý HTML dokument, ale pouze provedené změny.

Nevýhodou může být, že AJAX může naopak zvýšit počet vyměňovaných http požadavků, a třebaže přenáší nižší množství dat, při nevhodné implementaci zátěž neklesne. Problémem rovněž může být síťová latence. Potřeba komunikace přes Internet má negativní dopady na rychlost odezvy a interaktivitu uživatelského rozhraní. Pokud uživateli není signalizováno, že aplikace zpracovává jeho požadavek, může si myslet, že systém jeho požadavek ignoroval. Vhodným řešením je po dobu odbavení požadavku na server a jeho odpovědí uživateli zobrazit, že jeho požadavek se zpracovává, například textem nebo animovanou ikonou. [2] [3]

2.5.2 LINQ

Language Integrated Query je integrovaný dotazovací jazyk, jež byl poprvé uveden v .NET Frameworku 3.5. Již od začátku svého vývoje byl LINQ zamýšlen jako nástroj pro práci nad jakýmkoliv typem dat a zároveň zjednodušuje veškeré úkony nad nimi. [2]

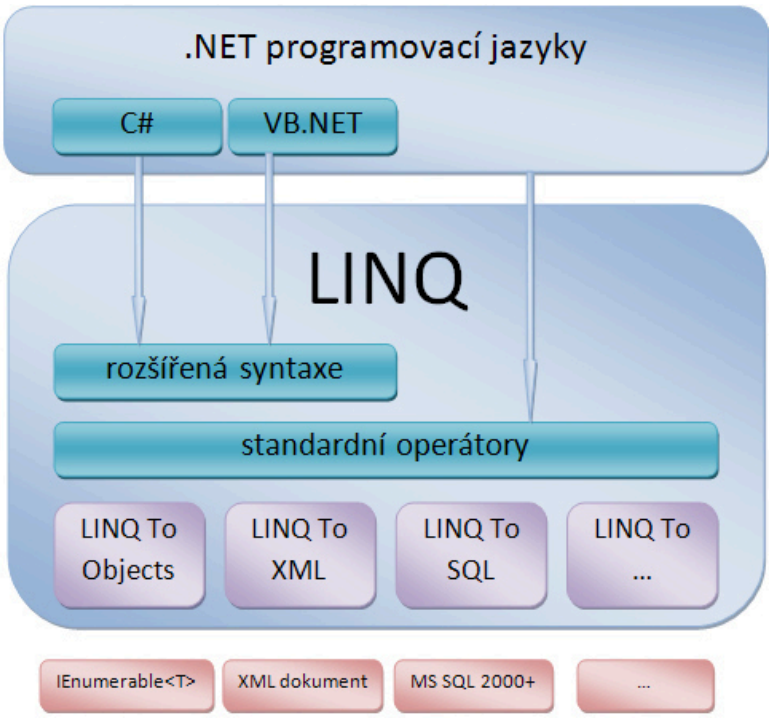
Linq je dostupný především pro tyto typy zdrojů:

1. **Linq to XML** – pracuje s daty v XML souborech bez využití standardních technik práce s XML soubory mezi které patří SAX (Sample API for XML) nebo DOM (Dokument Object Model), jedná se tedy o novou plnohodnotnou alternativu ve věci přístupu k XML datům.
2. **Linq to Objects** (Linq to DataSet) – práce nad daty v paměti Objects bývají kolekce nebo pole, DataSet kolekce dat získan´ych z databáze (technologie ADO.NET). Objekty implementují rozhraní: Ienumerable.
3. **Linq to SQL** – provádí dotazování nad databázemi, které používají rozhraní MS SQL Server. Linq příkazy jsou poté převedeny na SQL příkaz, kterýy je zpracován. Protože jsou data v těchto databázích uložena jako relační, je poté nasazen mapper těchto dat na objektová data. Objektová data pak používají LINQ. Tento princip je výhodný především v tom, že se s daty se pracuje jako s objekty. [2] [13]

Přestože LINQ to SQL neposkytuje žádnou funkcionalitu navíc, která by nešla duplikovat s kódem ADO.NET, jeho použití má několik nesporných výhod:

- Odstínění od rozdílů implementací SQL jazyků pro databáze různých výrobců (Linq sestavuje sám SQL dotazy).
- Není nutné psát kód ADO.NET (otevření spojení, uzavření spojení, …), čímž se kód zeštíhlí.
- Bez psaní kódu ADO.NET je možné změnit velké množství podrobnosti v datech, na které jsme se dotazovali a následně v jedné dávce potvrdit aktualizace. Toto je umožněno díky tzv. Defarred Execution – dotazy jsou prováděni dávkově, až je provedení nutné. [13]

Obr. 2.2 LINQ



Zdroj: http://www.zdrojak.cz/clanky/linq-a-lambda-expressions/

2.5.3 Entity Framework

Entity Framework (dále EF) je dynamický objektově-relační mapovací rámec. Jeho cílem je zajistit práci s daty (výběr, editace, přidání, odstranění, výpočty, sumarizace) v databázi. Databázové tabulky se přímo mapují na C# třídy, v kódu pracujeme jen s objekty a framework sám na pozadí generuje SQL dotazy. Vůbec se tedy nepracuje s jazykem SQL a aplikace se tak stává 100% objektová.

Entity Framework umožňuje navrhnout konceptuální schéma (EDMX diagram), které je automaticky převedeno do podoby databázové struktury včetně relací. Možný je samozřejmě i opačný postup převedení databázové struktury do EDMX schématu.

EDMX schéma uživatel vytváří v grafickém designéru a provedené změny jsou automaticky převáděny do zdrojového kódu používaného jazyka. Zdrojový kód je možné ručně doplňovat, či měnit, ale při opětovné změně v editoru bude kód znovu přegenerován a uživatelské změny mohou být ztraceny. [2]

2.5.4 TELERIK rozhraní pro ASP .NET MVC

Dle webových stránek společnosti Telerik je předním dodavatelm komponent uživatelského rozhraní pro Microsoft .NET technologie (ASP .NET AJAX, ASP .NET MVC, WinForms, Windows Presentation Foundation a Silverlight.

Telerik UI for ASP .NET MVC (dále Telerik UI) je kompletní framework pro budování moderních HTML5 webů a mobilních aplikací pomocí ASP .NET MVC. Obsahuje více než 70 grafických komponent (datové gridy, tlačítka, kalendáře, vstupy s automatickým doplňováním a mnoho dalších…). Díky jeho modularitě jej lze použít s funkcemi na straně klienta i na straně serveru. Nespornou výhodou je rovněž jeho podpora „vstřícného webdesignu“. V posledních letech se výrazně změnilo typické chování uživatelů, kdy stále častěji využívají tablety a chytré telefony. Telerik UI nabízí vývojáři flexibilní rozhraní, které je snadno přizpůsobitelné různým obrazovkám. [16]

2.5.5 Ninject

Ninject[2] je vkládač závislostí (Dependency Injection, dále jen DI). DI je návrhový vzor, který umožňuje vkládat závislosti mezi jednotlivými komponentami programu tak, aby mohla jedna komponenta používat druhou, aniž by na ni měla v době sestavování programu referenci. DI tedy souží pro snížení závislostí mezi jednotlivými částmi systému. Jeho provedení vychází z obecnějšího návrhového vzoru Inversion of Control (IoC). Výsledkem je větší kontrola nad objekty a tento návrh zároveň umožňuje lepší znovupoužitelnost. [2] [14]

Příklad použití Interface Injection (rozhraní v sobě obsahuje názvy metod, které předávají dané objekty) může výpadat následovně:

```
Public class Akce{
      public int Id { get; set; }
      …
}

public interface IakceRepository{
        Iqueryable<Akce> Akce { get; }
    }

public NinjectControllerFactory(){
      Ikernel ninjectKernel = new StandardKernel();
      ninjectKernel.Bind<IakceRepository>().To<EFAkceRepository>();}
```

[2] Ninject: http://www.ninject.org/

Pro entitu Akce vytvoříme rozhraní. To poté napojíme za pomocí Ninjectu na třídu obsahující implementaci repozitáře. Dále v aplikaci při potřebě volání entity Akce činíme přes rozhraní, které nám předá potřebný objekt.

Nevýhodu tohoto řešení může být menší přehlednost kódu a jeho náročnější správa.

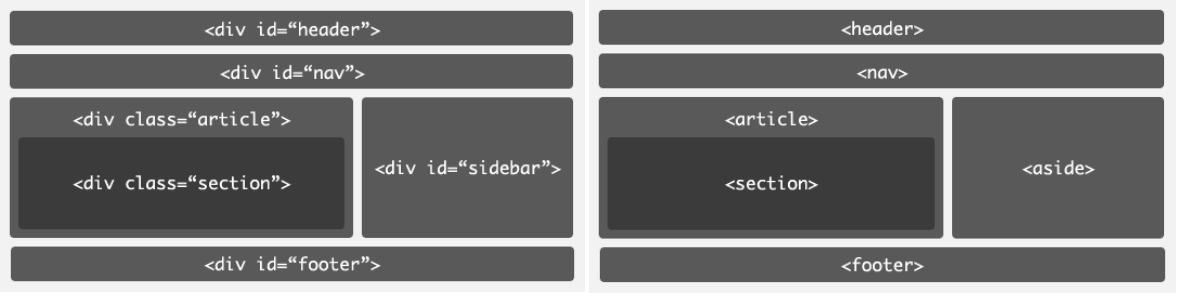
2.6 HTML5

HTML5 je nová verze specifikace značkovacího jazyka HTML, která přináší podstatné změny v technologiích webových stránek. HTML5 umožňuje kromě jiného přehrávat multimédia přímo ve webovém prohlížeči a vytvářet v něm aplikace, které fungují i bez připojení k internetu. V současnosti je vývoj HTML5 ve stádiu Candidate Recommendation.[3] Podle aktuálního plánu by měla být konečná specifikace HTML 5.0 schválena v posledním čtvrtletí roku 2014, její nástupnická verze HTML 5.1 v posledním čtvrtletí roku 2016[4] [8]

2.6.1 Strukturování webových stránek

HTML 5 uvádí celou sadu nových elementů, které významně usnadňují strukturování stránek. Většina současných stránek se sestává z ustálené skupiny obvyklých částí, jako je hlavička, patička a různé sloupce, přičemž se tyto části běžně definují pomocí elementů `div` odlišených prostřednictvím popisných atributů `id` a `class`. Takto se děje z důvodu, že specifikace HTML 4 postrádá nezbytné sémantické prvky pro přesnější vyznačení specifickcýh částí dokumentu. HTML 5 řeší tento problém zavedením nových elementů reprezentujících jednotlivé části dokumentu. Rozdíl strukturování dokumentu nám ukazuje obr. 2.3 kde v levé části je původní struktura v HTML 4 pomocí tagu `div` a v pravé části je ukázána struktura stránky pomocí nových elementů v HTML 5. [8]

Obr. 2.3 Rozdíl strukturování mezi HTML 4.1 a HTML 5



Zdroj: http://interval.cz/clanky/seznameni-s-html-5/

[3] HTML5: http://www.w3.org/TR/html5/
[4] http://dev.w3.org/html5/decision-policy/html5-2014-plan.html

Kód takového dokumentu by mohl vypadat následovně:

```
<body>
   <header>...</header>
   <nav>...</nav>
   <article>
      <section>
         ...
      </section>
   </article>
   <aside>...</aside>
   <footer>...</footer>
</body>
```

Bližší popis jednotlivých elementů:

- **Header** reprezentuje hlavičku sekce. Taková hlavička může obsahovat více než jen nadpis – například může být rozumné zahrnout do hlavičky podnadpis nebo informace o verzi obsahu.
- **Nav** reprezentuje sekci obsahující navigaci. Je vhodný pro navigaci celého webu i pro obsah dokumentu.
- **Aside** je určen pro obsah dotýkající se nějakým způsobem hlavního obsahu. Je vhodný pro vyznačování vedlejších sloupců.
- **Section** reprezentuje přirozené části dokumentu, jako jsou například kapitoly.
- **Article** reprezentuje nezávislou část dokumentu, stránky nebo webu. Je vhodný pro obsah typu zpráviček, blogpostů, uživatelských příspěvků nebo komentářů. [8]

2.6.2 Nové vlastnosti

HTML5 s sebou přináší kromě změny sémantiky rovněž spousty nových vlastností a tagů. Základním rozdílem je, že již není nutné uvádět verzi a odkaz na DTD (jak bylo dříve nutno u předchozích verzí HTML nebo XHTML). U HTML5 stačí prohlížeči říct, že se jedná o HTML. Základní !DOCTYPE definice tedy vypadá následovně: [17]

```
<!DOCTYPE html>
```

Výčet změn tagů a nových elementů (výše nepopsaných ve strukturování):

- **a** – HTML5 rozšiřuje možnosti tohoto tagu o možnost zabalení do hypertextového odkazu jakýchkoli i blokových elementů

- **b** – Tomuto elementu je nově v HTML5 přidělen zvláštní sémantický účel, a sice odlišení od zbytku textu. Na obsah elementu se neklade žádný zvláštní význam, text je pouze označen tučně.
- **canvas** – jde pouze o plátno pro dynamické vykreslování obrázků, grafiky, her, grafů atd. Element je tedy jen kontejner pro grafiku, pro skutečné vykreslení je nutné použít skript
- **cite** – HTML5 upřesňuje jeho využití pro označení citovaného titulku díla
- **details** – popis podrobností o dokumentu nebo části dokumentu
- **figcaption** a **figure** – figcaption je potomek elementu figure, který umožňuje vložení doplňkového textu například k obrázku nebo zdrojového kódu
- **scrip** – používá se pro seskupení nadpisů.
- **hr** – v HTML5 dostává sémantický účel představující „tematickou přestávku v textu“
- **input** – tento element se dočkal obsáhlého rozšíření v podobě podporovaných typů obsahu
- **mark** – představuje zvýraznění nebo označení určitého textu. Lze ho vyžít například pro zvýraznění vyhledávaných slov ve výsledku hledání.
- **meta** – u tohoto elementu došlo k pozměnění jeho atributů
- **small** – nový význam pro drobné poznámky nebo doplňky textu
- **summary** – pouze titulek, který se používá především pro prvek details
- **time** – element označuje časové údaje, nabízí nastavení vlastního formátu data a času [17]

2.7 CSS

HTML je omezeno jak z hlediska vzhledu, tak z hlediska technologie. HTML bylo totiž původně vymyšleno pouze k zobrazování informací. Cascading Style Sheets (Kaskádové styly) byly proto navrženy, aby pomohly s formátováním HTML specifikace. Stejně jako styly v textovém editoru, CSS poskytuje mechanismus, jímž lze snadno určit a měnit formátování, aniž by byl pozměněn původní kód. „Kaskády“ v názvu vychází ze skutečnosti, že specifikace umožňuje více stylům se ovlivňovat, což umožňuje jednotlivé webové dokumenty, které mají být formátovány, mírně odlišit od jejich příbuzných.

Styly lze deklarovat 3 způsoby:

- **řádkový (in-line) styl** – přímý zápis do formátované značky, nevýhodou je, že nepůsobí globálně.

```
<h1 style="font-weight: bold">Tento nadpis bude tučný. </h1>
```

- **vložením do hlavičky stránky (stylopis) –** Do hlavičky se vloží značka <style>, kde parametr type má hodnotu text/css.

  Do hlavičky dokumentu se napíše stylopis:

  ```
  <style> h1 {font-weight: bold;}</style>
  ```

  Do těla stránky se poté můžou zapsat odstavce:

  ```
  <h1>Tento nadpis bude tučný. </h1>
  <h1>Tento nadpis se zobrazí taktéž tučně. </h1>
  ```

- **Externím CSS souborem –** připojením externího souboru stylů, standardně s příponou .css. Připojení k HTML dokumentu se provádí v hlavičce použitím tagu: `<link rel="stylesheet" type="text/css" 21kri="styl.css">` V externím souboru se poté definují vlastnosti pro jednotlivé styly, které působí globálně na všechny elementy v dokumentu:

  ```
  h1 {font-weight: bold;}
  ```

2.7.1 CSS 3

Verze CSS 3 rozšiřuje specifikaci verze 2.1 o nové příkazy. Umožňuje přesněji stanovovat hodnoty parametrů a obsahuje také nové pseudotřídy a pseudoelementy pro přesnější přiřazení stylů. Přináší rovněž nové možnosti formátování textu. K dispozici jsou nové barevné profily. Nově lze definovat průhlednost, odstín, světlost, sytost a barevný tón barvy. [18]

Přestože je konečná specifikace CSS3 konsorciem W3C stále ve vývoji[5], je většina těchto vlastností podporována moderními prohlížeči.

2.8 JavaScript a jQuery

Jazyk JavaScript byl vytvořen ke konci minulého století společností Netscape. Jde o programovací jazyk, jehož činnost zabezpečuje prohlížeč webových stránek návštěvníka a ne server, na kterém jsou stránky uloženy. Charakteristikou je jazykem doplňkovým, neboť vše, co je v něm naprogramováno, nebude fungovat samostatně. Skript je vyvolán pouze tehdy, když tvoří součást HTML stránky, nebo se na něj HTML stránka přímo odvolá. JavaScript

[5] http://www.w3.org/Style/CSS/current-work

umožňuje doplnit do HTML prvky interaktivity (vypsání textu do stavového řádku, kontrola formulářů, efekty s prvky stránek a další). Pro účely označování začátku a konce skriptu v rámci stránky se požívá párová HTML značka `<script></script>`. [7]

Jelikož je JavaScript interpretován na straně klienta, má několik nevýhod. Z důvodu bezpečnosti (zneužití diskrétních informací, nebo mazání informací z disku klienta) jazyk nemá zabudovány funkce umožňující zapisovat a načítat údaje ze souborů. JavaScript také neumožňuje zpracovávat a centrálně uchovávat údaje týkající se provozu stránky. [7][7]

jQuery je rychlá a stručná JavaScriptová knihovna, která zjednodušuje zkřížení HTML dokumentu, zpracování událostí a Ajax interakce pro urychlení vývoje webových aplikací. jQuery většinou existuje jako jeden javascriptový soubor, obsahující všechny funkce. Lze jej stáhnout z oficiálních stránek *http://jquery.com/* a do webové stránky může byt vložen následovně [6]:

```
<script type="application/javascript" src="/cesta/jquery.js"></script>
```

Další způsob načítání jQuery je za pomocí využití sítí CDN (Content Delivery Network). Takto lze jQuery načíst v aktuální verzi přímo z domovské stránky projektu, nebo např. ze serverů Google:

```
<script type="application/javascript"
src="http://code.jquery.com/jquery-latest.min.js">
</script>

<script type="application/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/1.10.2/jquery.min.js">
</script>
```

2.9 UML

„Sjednocený modelovací jazyk (UML) je druh grafické notace podporovaný nezávislým meta-modelem, který umožňuje popisovat a navrhovat softwarové systémy, konkrétně systémy budované využitím objektově orientované (OO) metody.“ [1]

Jazyk UML má dle Fowlera tři hlavní využití při tvorbě aplikací. Nejčastějším je použití UML pro tvorbu náčrtku (UML as sketch), kde vývojáři v tomto pojetí používají UML k usnadnění komunikace o některých aspektech systému. Pokud se UML použije pro

[6] http://docs.jquery.com/Downloading_jQuery

podrobný návrh (UML as blueprint), bude jeho hlavní význam spočívat v kompletnosti. Posledním způsobem je použití UML jako programovacího jazyka. V takovémto vývojovém prostředí vytváří vývojáři UML diagramy, které jsou překládány přímo do spustitelného kódu a UML se stává zdrojovým kódem. [1]

2.9.1 UML diagramy

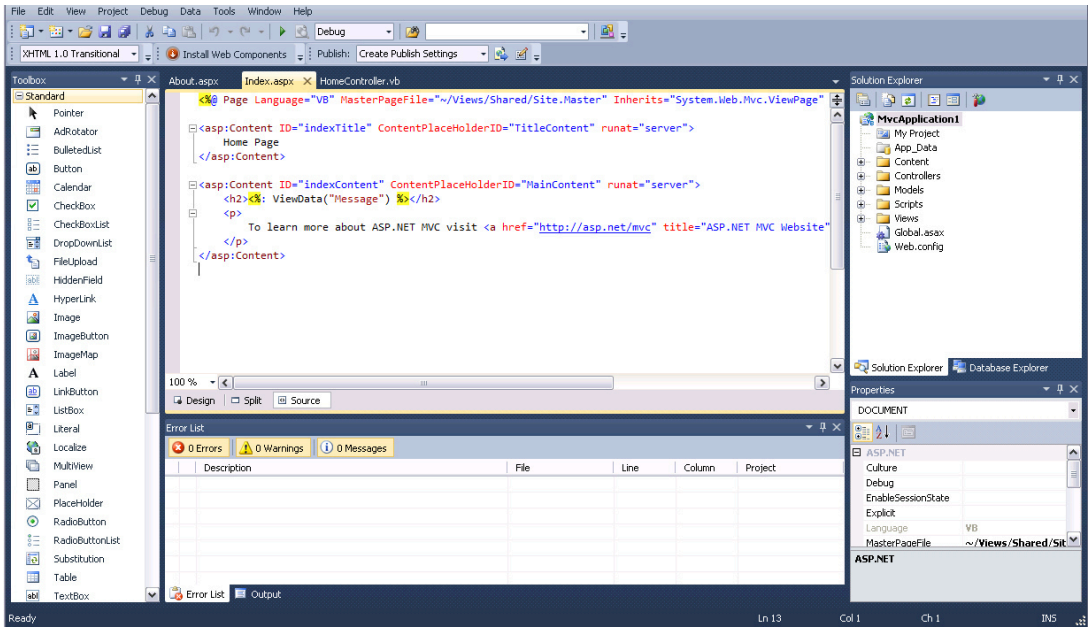
UML 2 popisuje 13 oficiálních typů diagramů. Následující tabulka nám uvádí jejich výčet a účel použití. [1]

Diagram	Účel	Původ
Aktivit	Procesní a paralelní chování	V UML 1
Balíčků	Hierarchická struktura kódu pro překlad	Neoficiálně v UML 1
Časování	Interakce mezi objekty; důraz na časování	Nový v UML 2
Komponent	Struktura a propojení komponent	V UML 1
Komunikace	Interakce mezi objekty; důraz na spojení	V UML 1 diagram spolupráce
Nasazení	Nasazení artefaktů na uzly	V UML 1
Objektů	Příklad uspořádání instancí	Neoficiálně v UML 1
Přehledu interakcí	Kombinace sekvenčního diagramu a diagramu aktivit	Nový v UML 2
Případu užití	Jak uživatelé komunikují se systémem	V UML 1
Sekvenční	Interakce mezi obbjekty; důraz na sekvence	V UML 1
Složených struktur	Dekompozice tříd za běhu programu	Nový v UML 2
Stavový	Jak události mění objekt během jeho života	V UML 1
Tříd	Třídy, vlastnosti a vztahy	V UML 1

2.10 Microsoft Visual Studio

Microsoft Visual Studio je profesionální vývojářský nástroj, který podporuje komplexní soustavu různých vývojových nástrojů, včetně sady nástrojů pro ladění a inteligentní doplňování kódu. Inteligentní doplňování kódu urychluje proces kódování aplikací tím, že snižuje nedorozumění, překlepy a další běžné chyby. Toto je prováděno pomocí vyskakovacích oken s automatickým dokončením při psaní, dotazování parametrů funkcí a dotazovacích rad u syntaktických chyb. Ve Visual Studiu se tento nástroj jmenuje IntelliSense. Visual Studio má rovněž zabudovaný server určený pro testování. [5]

Obr. 2.4 Vývojové prostředí Microsft Visual Studio



Zdroj: Microsoft Visual Studio 2010

Microsoft Visual Studio je dostupné v několika verzích. Pro tvorbu webové aplikace byla využita verze Visual Studia 2010.

Výhody a přednosti Visual Studia:

- **Integrovaný webový server –** Do prostředí Visual Studia je integrován testovací webový server, jež umožňuje spouštět webové aplikace přímo z vývojového prostředí.
- **Vývoj ve více jazycích –** Visual Studio umožňuje psát kód buď v jednom jazyku, nebo v několika rozdílných jazycích a přitom používat stále stejné vývojové rozhraní (IDE). Rovněž umožňuje programovat jednotlivé webové stránky v různých jazycích a začlenit všechny to téže webové aplikace.
- **Intuitivní styl psaní kódu –** Visual Studio pomáhá vývojáři při psaní kódu automatickým formátováním, odsazování a používáním různých barev pro odlišení různých částí kódu.
- **Rychlejší vývoj –** Mnohé ze schopností Visual Studia jsou zaměřeny pro rychlejší práci. Schopnosti týkající se pohodlí, jako například výkonné vyhledávání a nahrazování či automatické přetváření kódu na komentáře, nebo již zmiňované IntelliSense, umožňují pracovat rychle a efektivně.
- **Ladění –** Visual Studio obsahuje nástroje určené pro ladění. S jejich pomocí lze snadno vystopovat chyby a diagnostikovat podivné chování. Kód lze vykonávat řádek po řádku,

nastavovat inteligentní body přerušení (breakpoints). Rovněž lze prohlížet informace nacházející se v paměti počítače. [5]

2.11 Charakteristika společnosti On line System

Společnost On line system s.r.o. vznikla v roce 1997. Současně se vznikem divize webdevelopmentu začala v roce 1999 vznikat divize měření pro sportovní aktivity. V roce 2006 přibyla divize TV grafiky pro významné sportovní projekty. Společnost měří rozličné druhy sportovních aktivit prostřednictvím japonské čipové technologie Jchip. Mezi největší závody, kde společnost zajišťuje služby časomíry patří běžecké seriály RUN.CZECH (viz http://www.runczech.com) a Run-Tour (viz http://www.run-tour.cz). Kromě těchto běžeckých závodů firma rovněž zajišťuje služby časomíry dalšímu množství rozličných závodů, ať již běžeckých, cyklistických nebo inlinových. Rovněž zajišťuje měření atletických mítinků, kde kromě české atletické extraligy (viz. http://www.atletika.cz) měří i různé mezinárodní soutěže. Mezi nejznámější lze zařadit Zlatou tretru v Ostravě.

Kromě činností v oblasti měření sportovních událostí je firma od roku 2005 hlavním distributorem výrobků TIMETRONICS (cílové kamery, systém chybného startu, větroměry, optické měřiče vzdáleností atd.) pro Českou republiku, Slovensko a Polsko.

Firma se rovněž zabývá vývojem zakázkového software. Společnost poskytuje tyto služby v oblasti internetových aplikací, sportovních soutěží, multimédií, designu a televizní grafiky.

Obr. 2.5: Logo společnosti On line System

3 Analýza současného stavu výsledkového servisu

Společnost On line System, pro kterou je webová aplikace vyvíjena, již vlastní webové stránky pro prezentaci výsledků měřených sportovních událostí, ale tyto ztratily svou funkčnost s přechodem na nový software pro měření závodů Onlinesport (vyvíjen vlastním programátorským týmem), který má rozdílně definovány základní databázové struktury. V současné době je tudíž výsledkový servis nefunkční a obsahuje pouze archiv výsledků starších závodů měřených původním měřícím softwarem Maratonec.

Naměřené výsledky jsou v současnosti zpravidla k dispozici k nahlédnutí ve vytištěné podobě v místě konání závodu (pouze omezenou dobu po skončení závodu). Dále jsou výsledky zasílány organizátorovi závodu ve formátu PDF nebo XLS, který tyto umisťuje na oficiální stránky závodu (pokud je zde tato možnost). Rovněž výsledky zpravidla zveřejňují různé sportovní a běžecké weby a online magazíny (např. Behej.com, BezvaBeh.cz…). V případě sportovních událostí pod hlavičkou Českého atletického svazu jsou výsledky rovněž k dispozici na jejich oficiálních stránkách *www.atletika.cz*.

3.1 Formální specifikace požadavků na nový systém

Pod pojmem požadavek se rozumí popis nebo specifikace jisté funkce nebo vlastnosti, kterou by měla webová aplikace disponovat. Požadavky lze rozlišit na dva základní typy, a sice *funkční* – specifikují požadavky na funkčnost webové aplikace a *nefunkční* – specifikují jiné vlastnosti webové aplikace, nebo podmínky omezující fungování webové aplikace.

3.1.1 Funkční požadavky

- **Kalendář závodů –** Webová aplikace bude zobrazovat kalendář závodů, u nichž bude společnost On line System zajišťovat služby časomíry, jsou schválené (je potvrzeno jejich měření oboustrannou dohodou mezi organizátorem a společností On line System) a společnost je chce zobrazit na těchto stránkách. Pro databázi závodů bude využita existující interní databáze pro systém administrace.
- **Výsledky závodů –** Aplikace bude obsahovat přehledné zobrazení výsledků změřených závodů. Kromě celkových výsledků bude možné vybírat výsledky podle jednotlivých tratí a kategorií. Samozřejmostí je řazení a pokročilé filtrování.

- **Export výsledků do xls –** Aplikace umožní uživatelům exportovat vybrané výsledky z výsledkové tabulky do formátu xls.
- **Oficiální výsledky –** Aplikace umožní uživatelům stáhnout oficiální výsledkové listiny (s hlavičkou závodu) ve formátu vydané společností. Tyto výsledky jsou ve formátu PDF.

3.1.2 Nefunkční požadavky

- **Implementace –** Webová aplikace je implementována pomocí platformy ASP .NET MVC s využitím TELERIK Kendo UI for ASP .NET MVC pro zobrazování kalendáře akcí a výsledkových tabulek.
- **Data –** Správu dat zajišťuje databázový server MS SQL, který je umístěn na firemním serveru.
- **Výstup –** Výstup webové aplikace je ve formátu HTML 5 a formátován pomocí CSS, tak aby se korektně zobrazoval v prohlížečích Mozilla Firefox, Google Chrome, Internet Explorer 8 a vyšší a mobilních prohlížečích.

3.2 Specifikace uživatelského vzhledu

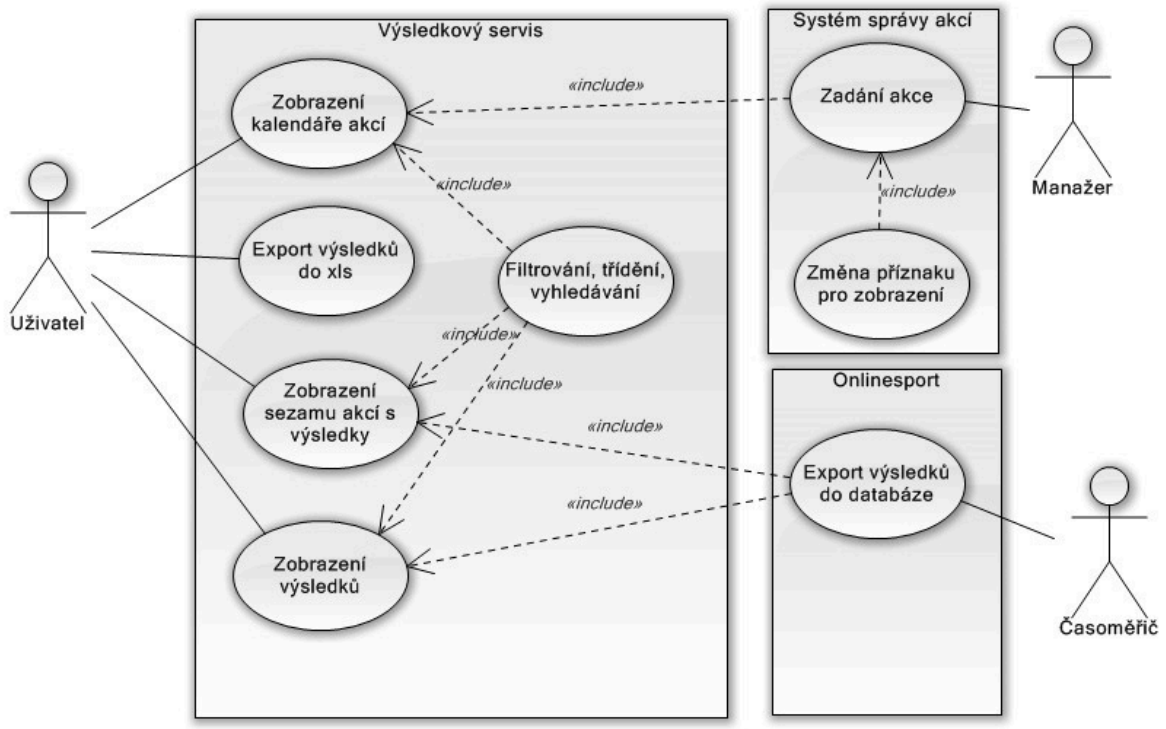
Uživatelský vzhled by měl být v souladu s barvami, které společnost využívá pro své aktivity. Typickými barvami jsou oranžová a modrá. Kromě podpory desktopových prohlížečů by stránky měly rovněž podporovat mobilní prohlížeče, respektive mít flexibilní vzhled, který se těmto prohlížečům přizpůsobí ve smyslu rozdílného uživatelského rozhraní, korespondující s omezeným prostorem displeje těchto zařízení. Celkový vzhled by měl být založen na jednoduchosti, přehlednosti a uživatelské přívětivosti, tak aby navigace a používání veškerých funkcí bylo zřejmé a uživatel je nemusel složitě vyhledávat, případně je vůbec neobjevil.

3.3 Analýza

Následující analýza objasňuje podstatu problému výsledkového servisu sportovních událostí. Na základě výše specifikovaných požadavků bude vytvořen Use Case Diagram, který definuje procesy prováděné uživatelem aplikace, a dále je zde naznačena interakce aplikace s dalšími systémy a uživateli, jež společnost využívá.

3.3.1 Use case diagram

Obr. 3.1: Use Case diagram



Zdroj: Vlastní zpracování v programu Software Ideas Modeler

3.3.2 Případy užití

Tabulka 3.1: Případ užití zobrazení kalendáře akcí

Případ užití: Zobrazení kalendáře akcí
ID: UC1
Účastníci: Uživatel
Vstupní podmínky: Akce je vytvořená v Systému správy akcí Akce má příznak zobrazení na webu Ano
Typický průběh: Systém zobrazí kalendář akcí v podobě přehledné tabulky, ve které je možno filtrovat, seřazovat a vyhledávat.
Následné podmínky:
Alternativní tok událostí:

Tabulka 3.2: Případ užití zobrazení seznamu akcí s výsledky

Případ užití: Zobrazení seznamu akcí s-výsledky
ID: UC2
Účastníci: Uživatel
Vstupní podmínky: Akce je vytvořená v Systému správy akcí Akce má nahrány výsledky Akce má příznak zobrazení na webu Ano
Typický průběh: Systém zobrazí seznam akcí s výsledky v podobě přehledné tabulky, ve které je možno filtrovat, seřazovat a vyhledávat.
Následné podmínky:
Alternativní scénář: V případě Akce typu atletika, která není měřena prostřednictvím softwaru Onlinesport, systém zobrazí detaily akce s odkazem na web atletika.cz, kde uživatel nalezne výsledky.

Tabulka 3.3: Případ užití zobrazení výsledků akce

Případ užití: Zobrazení výsledků akce
ID: UC3
Účastníci: Uživatel
Vstupní podmínky: Akce je vytvořená v Systému správy akcí Akce má nahrány výsledky Je vybrána akce ze seznamu akcí s výsledky Akce má příznak zobrazení na webu Ano
Typický průběh: Systém zobrazí výsledky vybraného závodu v podobě přehledné tabulky, ve které je možno filtrovat, seřazovat a vyhledávat.
Následné podmínky:
Alternativní tok událostí:

3.4 Datová analýza

Účelem této analýzy je vytvoření modelu budoucí webové aplikace. Nejprve byla prozkoumána stávající databáze, která již obsahuje tabulky pro akce, zda obsahuje veškeré potřebné atributy pro zajištění správné funkčnosti kalendáře akcí budoucí aplikace. Analýzou vyplynula potřeba rozšíření tabulky *Akce* o příznak pro potvrzení zobrazení na webu. Společnost si nepřeje zveřejňovat veškeré akce (některé události slouží pro interní účely) a běžnými způsoby by nebylo možné provést vyfiltrování těchto událostí.

3.4.1 Lineární zápis datové základny

Lineární zápis popisuje entity a vazby prostřednictvím klíčových atributů. Nejprve je uveden název tabulky v databázi a v závorce výpis klíčových atributů pro aplikaci. Tučně je uveden primární klíč, zvýraznění podtržením značí cizí klíč.

Akce (**Id**, DatumOd, DatumDo, TypSportuId, Nazev, Mesto, Misto, Serial, StavAkceId, Web, WwwOnline)

CisDetail (**Id,** Kod, Nazev)

WwwCategory (**Id**, IdTrack, NameLong, NameShort)

WwwDocuments (**Id**, IdTrack, DocumentName, DocumentLink)

WwwRacerStatus (**Id,** Status)

WwwRacer (**Id**, IdEvent, IdKategorie, Status, BIB, Name, Club, Result, ResultReal, Sate, Sex, TeamName, OrderTeam, OrderAbs, OrderCat, OrderSex, Born, OrderCat2, OrderCat3)

WwwRacerSplit (**Id**, IdSplit, IdRacer, Time)

WwwSplit (**Id**, IdTrack, Name)

WwwTrack (**Id**, IdEvent, Name)

Tabulky Akce a CisDetail již databáze obsahuje, zbývající tabulky budou pro účely aplikace vytvořeny. Nové tabulky budou vytvořeny s předponou Www, která byla požadována pro odlišení nových tabulek v databázi sloužících pro výsledkový servis. Databázová struktura byla vytvořena na základě spolupráce s interním programátorem a požadavky na její správnou strukturu pro korektní zobrazování výsledků a možný import dat ze sowftware pro zpracování výsledků Onlinesport.

3.4.2 Vztahy mezi entitami

Následující výpis popisuje vztahy mezi jednotlivými entitami aplikace.

ObsahujeZávodníky (*Akce, WwwRacer* – 1:N) – akce (*Akce*), v případě, že již byl proveden export výsledků obsahuje závodníky (*WwwRacer*).

MáTrať (*Akce, WwwTrack* – N:1) – Akce (Akce) může mít přiřazenou trať (*WwwTrack*).

ObsahujeKategorie (WwwTrack, WwwCategory – 1:N) Trať (WwwTrack) obsahuje kategorie (WwwCategory).

MáKategorii (WwwRacer, *WwwCategory* – N:1) – každý závodník (*WwwRacer*) má přiřazenu Kategorii (*WwwCategory*) na základě pohlaví, věku…

MáStatus (*WwwRacer, WwwRacerStatus* – N:1) – každý závodník (*WwwRacer*) má přiřazen status (*WwwRacerStatus*) na základě jeho dosaženého výsledku v závodě.

TraťMáMezičasy (*WwwTrack*, *WwwSplit* – 1:N) – každá trať (*WwwTrack*) může mít několik mezičasů (*WwwSplit*).

ZávodníkMáMezičasy (*WwwRacer*, *WwwRacerSplit* 1:N) – závodník (*WwwRacer*) může mít naměřeno několik mezičasů (*WwwRacerSplit*).

MáČasy (*WwwSplit*, *WwwRacerSplit* 1:N) – mezičas (*WwwSplit*) obsahuje naměřené časy závodníků (*WwwRacerSplit*).

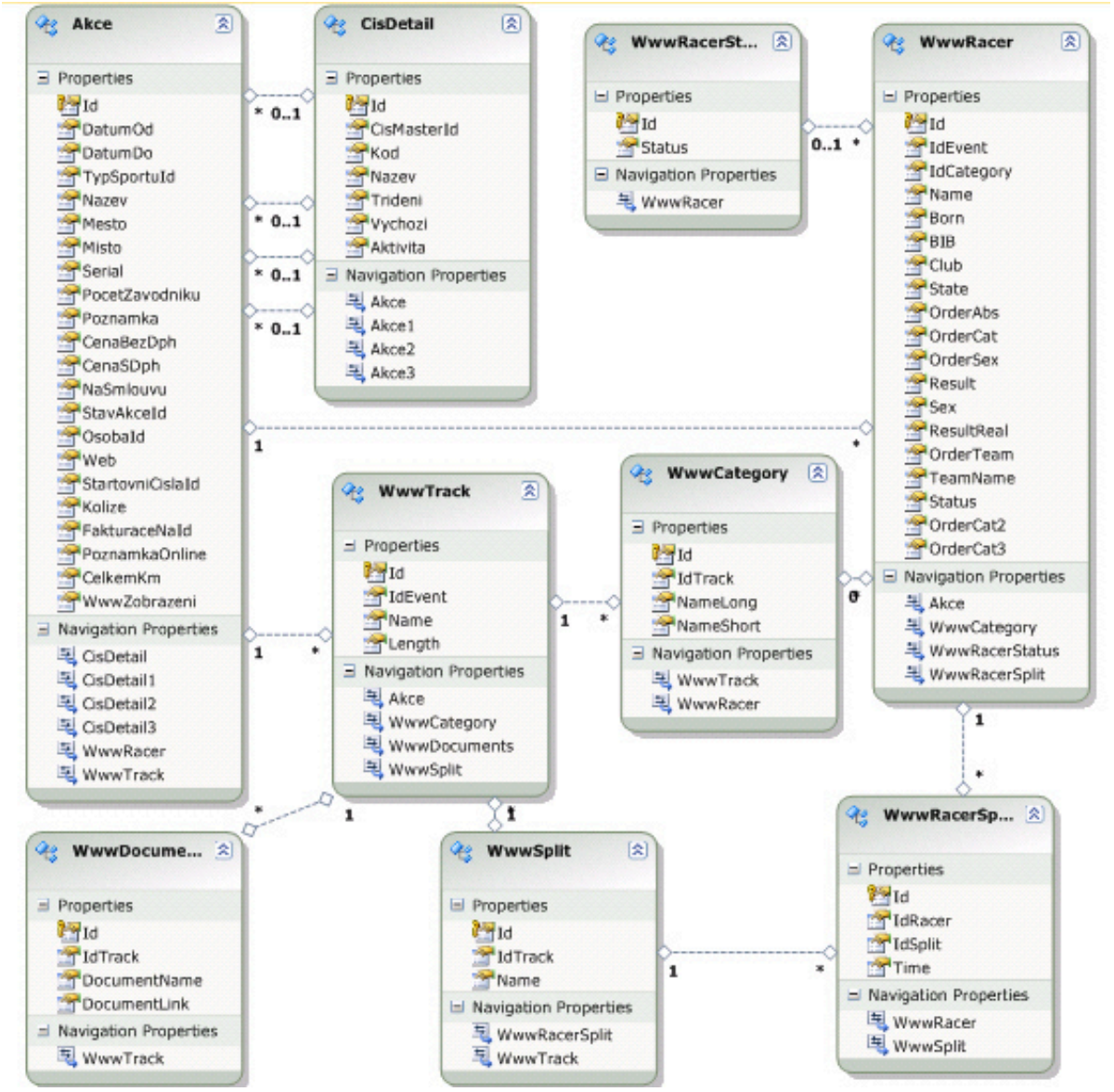
ObsahujeDokumenty (*WwwTrack*, *WwwDocuments* 1:N) každá trať (*WwwTrack*) může obsahovat několik dokumentů (*WwwDocuments*).

MáČíselník (*Akce*, *CisDetail* – N:N) – Akce (*Akce*) má číselník (*CisDetail*) pro potřebné sloupce.

3.4.3 ER diagram

ER diagram na konceptuální úrovni abstrakce popisuje uživatelskou aplikaci za účelem specifikace budoucí struktury databáze.

Obr. 3.2 ER diagram



Zdroj: Microsoft Visual Studio 2010

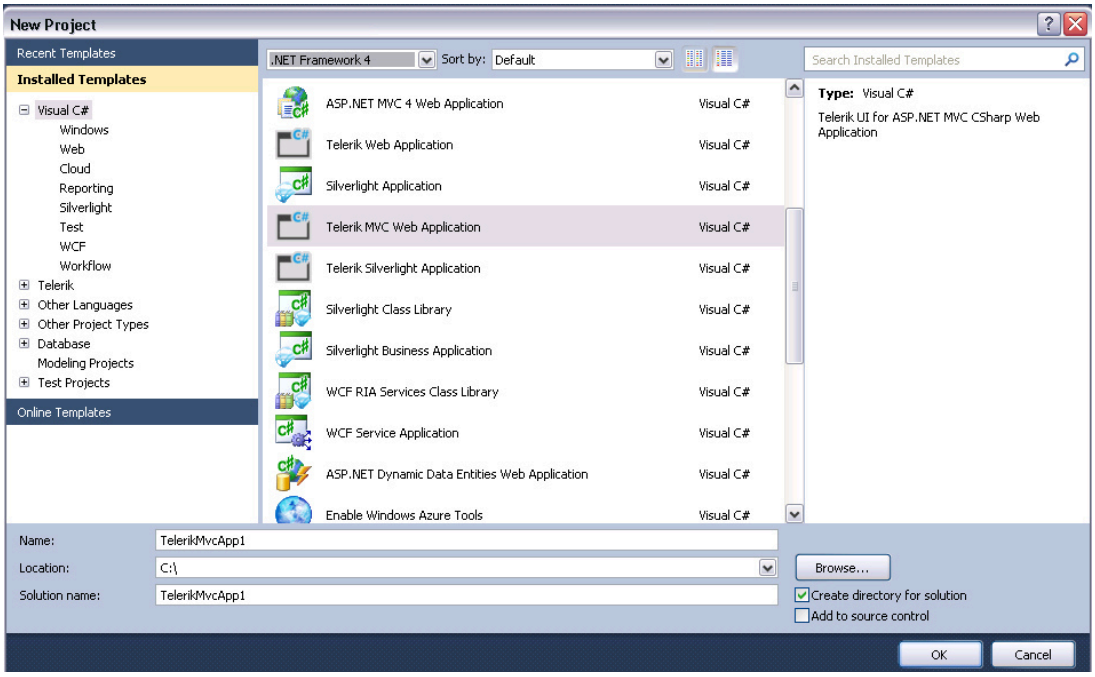
4 Návrh a realizace systému

Tato část diplomové práce se zabývá návrhem a implementací webové aplikace výsledkového servisu. Popisuje základní kroky a prvky použité při vývoji, charakteristiky jednotlivých formulářů a seznamuje čtenáře se základním vzhledem webové aplikace.

4.1 Implementace TELERIK UI pro ASP .NET MVC do Visual Studia

Jelikož jedním z požadavků na novou aplikaci bylo využití uživatelského rozhraní společnosti Telerik, bylo pro jeho použití nutné nejprve implementovat tento framework do samotného Visual Studia. Samotná instalace je velmi jednoduchá a provádí se prostřednictvím Telerik Control Panelu, kde je možné vybrat potřebné komponenty k instalaci. Pro náš projekt plně dostačuje TELERIK UI for ASP .NET MVC. Po provední instalace se nový projekt nezakládá jako klasický ASP .NET MVC, ale jako Telerik MVC Web Application (viz. Obr. 4.1). Je rovněž možné založit projekt jako obyčejnou MVC 4 aplikaci, ale v tomto případě by bylo nutno ručně přidávat a kopírovat veškeré referenční skripty a knihovny, případně využít CDN odkazy[7].

Obr. 4.1 Vytvoření nové Telerik MVC webové aplikace



Zdroj: Microsoft Visual Studio 2010

Poté co je vytvořen nový projekt jsou jednotlivé komponenty uživatelského rozhraní používány pomocí helperu `@(Html.Kendo().`*`Komponenta()`*`.`*`ParametryKomponenty`*`)`.

[7] Kendo UI: http://docs.telerik.com/kendo-ui/getting-started/using-kendo-with/aspnet-mvc/asp-net-mvc-4

4.2 Vrstva datového přístupu

Vrstva datového přístupu slouží ke komunikace s datovým úložištěm. Datovým úložištěm je databáze provozovaná na databázovém serveru Microsoft SQL Server 2008 umístěná na firemním serveru. Třídy této vrstvy neobsahují aplikační logiku. Jelikož aplikace nevyužívá administrační rozhraní – potřebná administarce je prováděna přes jiný, již existující aministrační systém, implementují tyto třídy pouze operace pro čtení objektů. Aplikační vrstva přistupuje k datům prostřednictvím definovaných rozhraní. Jsou zde definována dvě rozhraní. První zajišťuje přístup k entitě Akce, druhé zabezpečuje přístup k entitám potřebným pro zpracování a zobrazení výsledkových tabulek (WwwRacer, WwwCategory, WwwTrack, WwwRacerStatus).

Pro navázání spojení s databází je využita knihovna Entity Framework, která bývá nejčastěji využívána pro funkce související s objektově-relačním mapováním. Rovněž umožňuje automatické vytváření databázových spojení dle údajů zadaných v konfiguračním souboru Web.config.

4.3 Struktura aplikace

Adresářová struktura aplikace odpovídá konvencím aplikačního rámce ASP .NET MVC 4. Je tedy koncipována tak, aby promítala architekturu Model-View-Controller. Aplikace je rozdělená na 2 projekty. **123Online.Domain**, který obsahuje jednotlivé doménové entity a logiku nastavení pro perzistenci prostřednictvím repozitáře vytvořené pomocí Entity Frameworku, druhý projekt **123Online.WebUI** zahrnuje Controllery a View a funguje jako rozhraní pro 123Online aplikaci. Tento projekt kromě typických složek pro MVC architekturu obsahuje některé další. Podrobnější popis a struktura obsahu těchto projektů je uvedena níže.

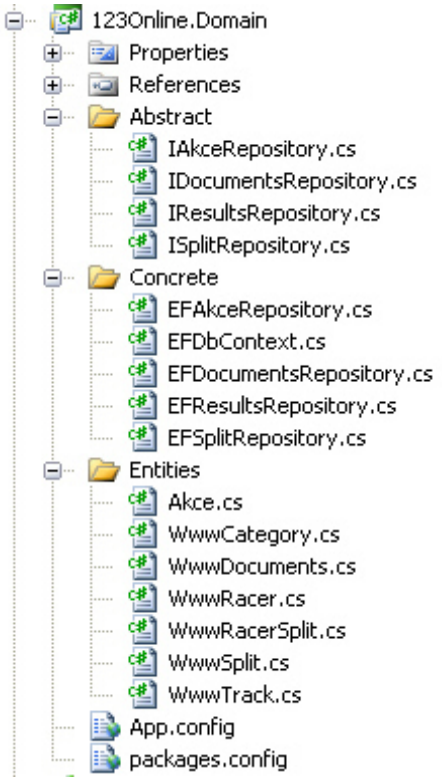
4.3.1 123Online.Domain

Tento projekt tvoří model celé aplikace. K oddělení modelu od uživatelského prostředí bylo přistoupeno především z důvodu možného budoucího rozšíření aplikace. Účelem tohoto projektu je zprostředkování přístupu do databáze a správa entit. Tento projekt má následující strukturu:

- **123Online.Domain.Abstract –** Tato část projektu obsahuje rozhraní zprostředkovávající komunikaci s databází

- **123Online.Domain.Concrete –** V této části dochází k implementaci výše zmíněných rozhraní (např. EFAkceRepository – předpona „EF“ značí využití Entity Frameworku)
- **123Online.Domain.Entities –** Jak je z názvu patrné, tato část obsahuje jednotlivé entity, které jsou používány pro komunikaci s databází.

Obr. 4.2 Adresářová struktura jmenného prostoru 123Online.Domain



Zdroj: Microsoft Visual Studio 2010

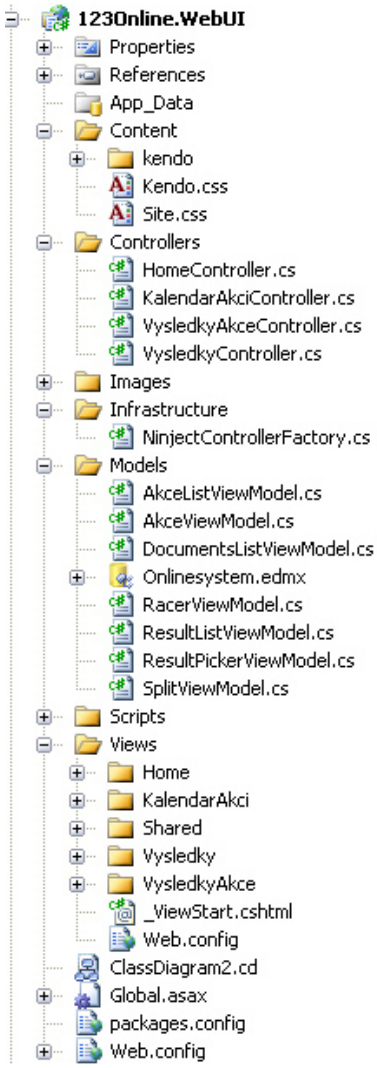
4.3.2 123Online.WebUI

Jak již název napovídá, tento projekt se zaměřuje na samotné uživatelské rozhraní aplikace. Struktura aplikace odpovídá paradigmatu *convention over configuration* – konvence nad konfigurací, na který ASP .NET MVC framework spoléhá.

- **123Online.WebUI.Content** – Složka obsahuje soubory s kaskádovými styly pro definici vzhledu aplikace a vzhled Telerik Kendo UI.
- **123Online.WebUI.Controllers** – Obsahuje veškeré Controllery aplikace. Controllery jsou členěny v závislosti na částech aplikace, které zpracovávají.
- **123Online.WebUI.Images** – Obrázky hlavičky, log závodů a ikon druhů sportů pro kalendář akcí a výsledky.
- **123Online.WebUI.Infrastructure** – Obsahuje třídu NinjectControllerFactory, která zabezpečuje předávání závislostí (Dependency Injection).

- **123Online.WebUI.Models** – Obsahuje View Modely pro jednotlivé zobrazované entity. Třídy téměř totožně kopírují výchozí datový model, jsou ovšem přizpůsobené pro potřeby zobrazování na stránkách rozšířením o další potřebné vlastnosti. Reference na datovou vrstvu je dána prostřednictvím principu Dependency Injection.
- **123Online.WebUI.Scripts** – Složka obsahuje jQuery knihovny a knihovny pro Telerik Kendo User Interface zabezpečující jeho funkcionalitu.
- **123Online.WebUI.Views** – Tento jmenný prostor obsahuje veškeré Views obsažené v aplikaci. Tyto jsou dále rozděleny do složek podle jednotlivých Controllerů, které tyto View obsluhují. Pro tvorbu všech View je využit jazyk HTML 5 a šablonovací systém Razor.

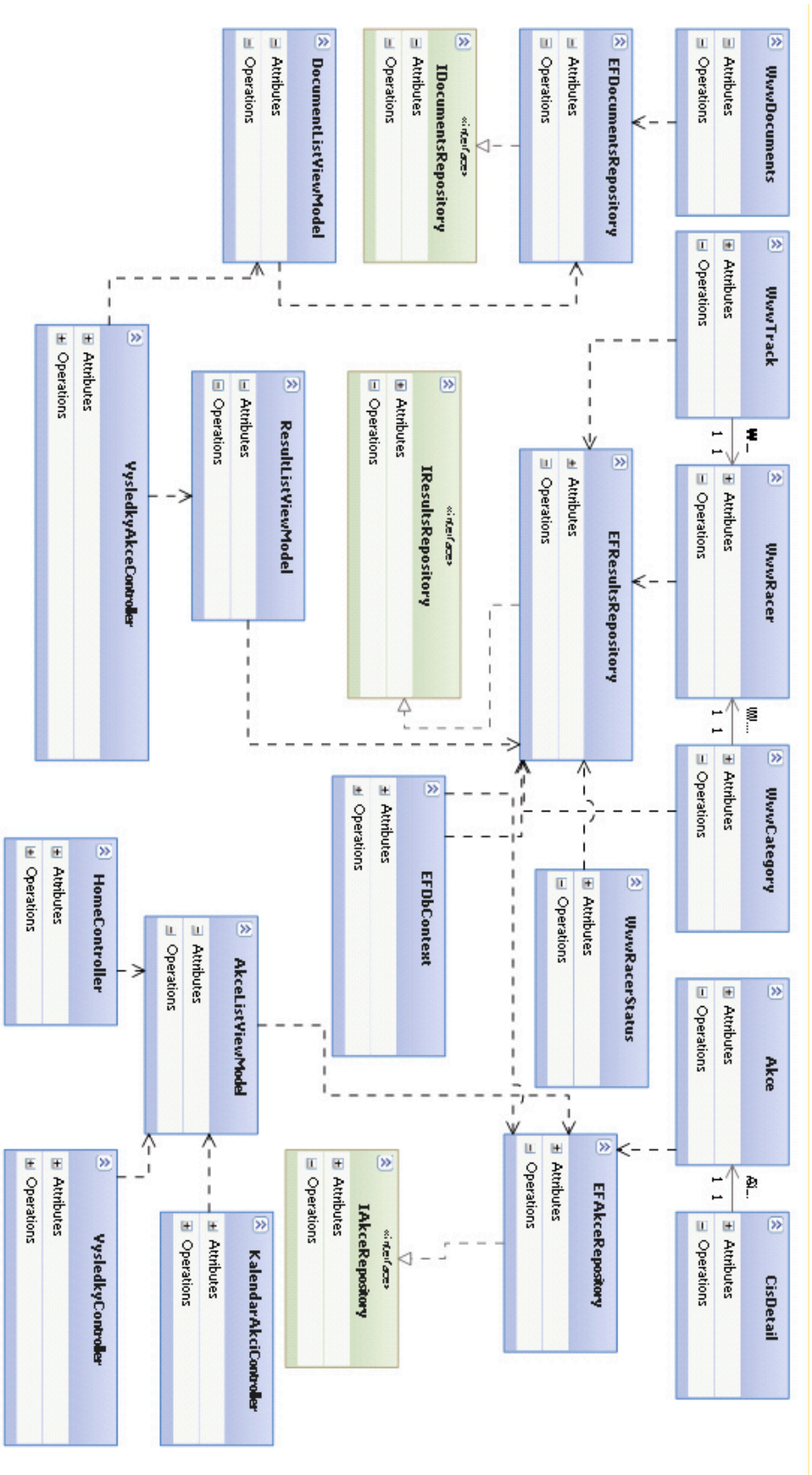
Obr. 4.3 Adresářová struktura jmenného prostoru 123Onlline.WebUI



Zdroj: Microsoft Visual Studio 2010

4.4 Diagram Tříd

Obr. 4.4 Diagram tříd



Zdroj: Microsoft Visual Studio 2010

Obrázek 4.4 zobrazuje diagram tříd aplikace, využívající architektovnický vzor Model-View-Controller. Data z databáze se získávají na základě vzoru „repozitář“. Z prostorových důvodů neobsahuje veškeré třídy, ale lze z něj vyčíst jakým způsobem jsou jednotlivé třídy propojeny a jaké jsou mezi nimi závislosti.

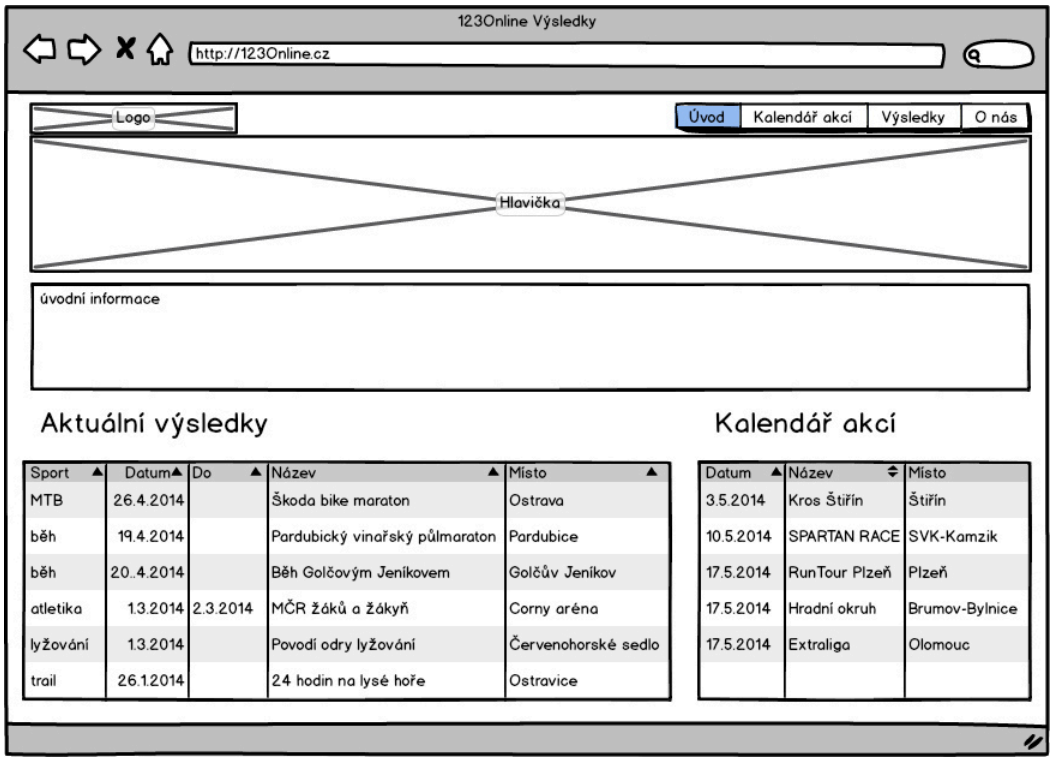
4.5 Vzhled a uživatelské rozhraní aplikace

Uživatelské rozhraní je jednou z nejdůležitějších částí webové aplikace. Při návrhu uživatelského rozhraní byl kladen důraz na jednoduchost a přehlednost ovládání. Ovládací prvky byly navrženy s ohledem na mobilní dotyková zařízení.

Při podpoře mobilních zařízení je rovněž nutné počítat s rozdílným rozlišením. Aplikace je navržena tak, aby se automaticky přizpůsobovala rozdílným velikostem zobrazovacích zařízení. K docílení tohoto stavu je využit příkaz `media`, který je definován ve standardu W3C CSS3[8]. V případě zařízení s nižším rozlišením než 850 pixelů se využije alternativní vzhled. Příklad použití tohoto příkazu vypadá následovně:

```
@media only screen and (max-width: 850px) { … }
```

Obr. 4.5: Návrh vzhledu úvodní stránky webové aplikace



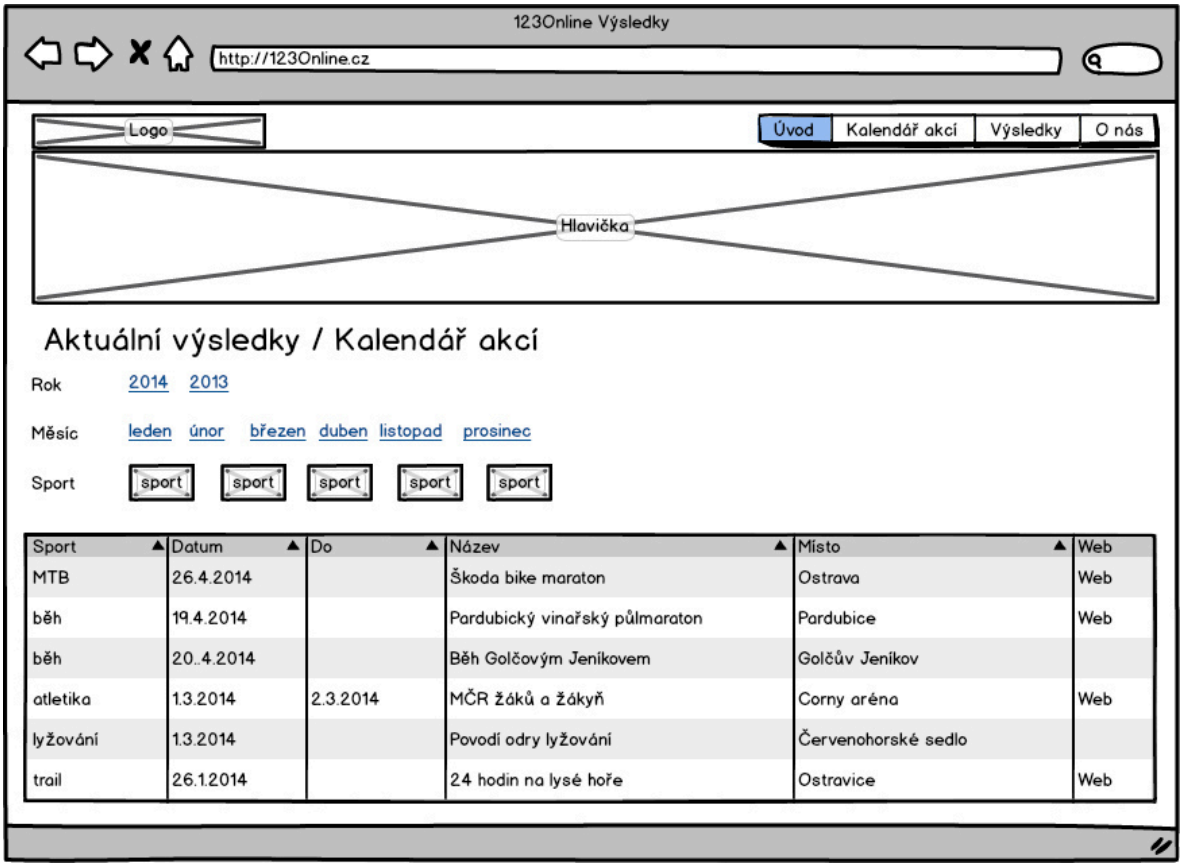
Zdroj: Vlastní zpracování v programu Balsamiq Mockups for Desktop

[8] CSS3 media: http://www.w3.org/TR/css3-mediaqueries/

Obrázek 4.3 ukazeje návrh (mockup) vzhledu úvodní stránky aplikace. Menu se nachází v pravé horní části aplikace. V případě mobilních zařízení se toto přesune na střed stránky z důvodu lepší přístupnosti. Pod hlavičkou (úvodním obrázkem) a sekcí s úvodními informacemi se nachází tabulky pro výčet prvních několika záznamů akcí s aktuálními výsledky a kalendáře akcí. Při použití mobilního prohlížeče se pro lepší použitelnost Kalendář akcí přesune pod aktuální výsledky. Flexibilita Telerik rozhraní zajistí roztažení těchto prvků (Aktuální výslekdy, Kalendář akcí) na celou šířku obrazovky v závislosti na použitém rozlišení.

Na obrázku 4.6 je zobrazen návrh vzhledu pro sekce Kalendář akcí a Aktuální výsledky. Horní část zůstala nezměněná (viz. 4.5.1). Pod nadpisem sekce se nachází menu pro výběr dle konkrétních kritérií, a sice dle roku, měsíce nebo sportu. Tato navigace je generována podle dat vyčtených v databázi, nelze tedy vybrat rok, měsíc či sport, které neobsahují žádné data.

Obr. 4.6: Návrh vzledu Kalendáře akcí a Aktuálních výsledků

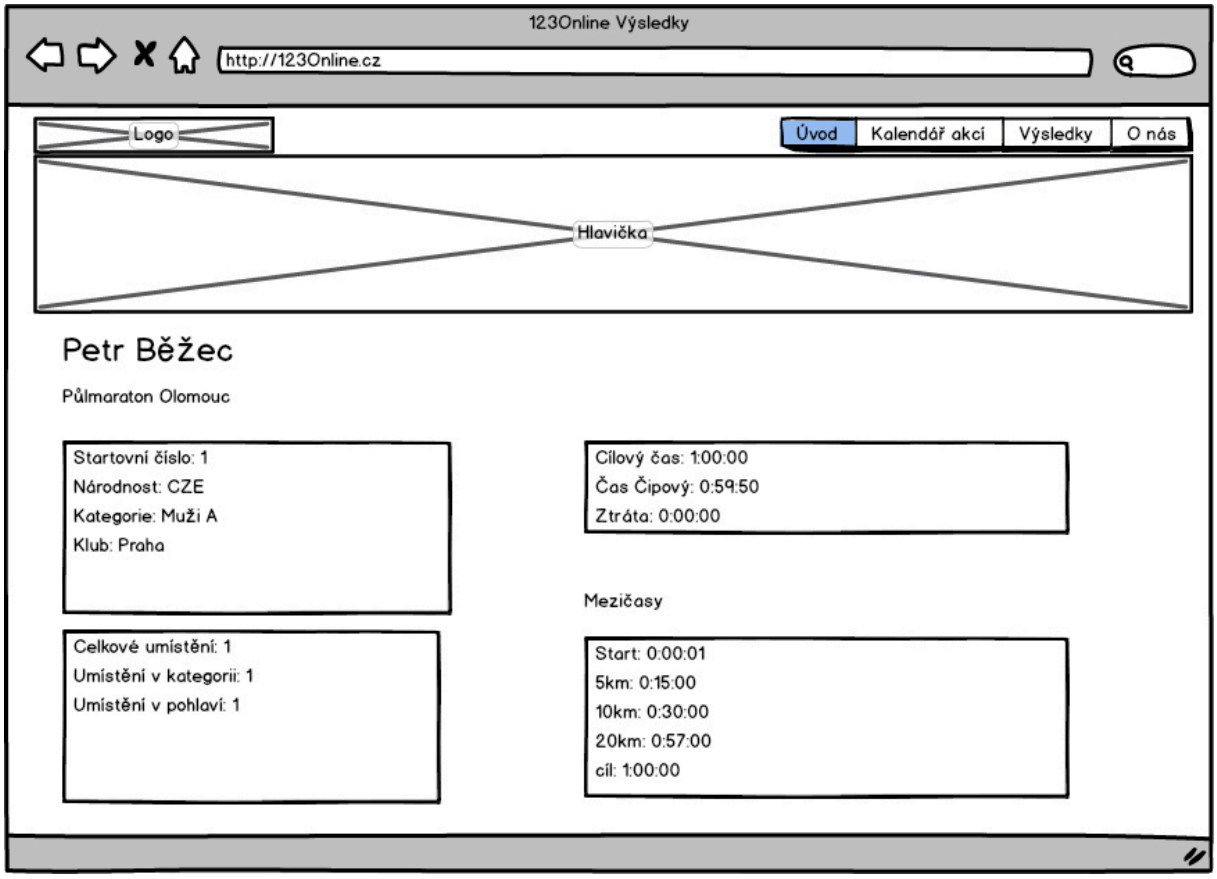


Zdroj: Vlastní zpracování v programu Balsamiq Mockups for Desktop

Stránka pro výsledky konkrétního závodu bude koncipována na podobném principu. Změna bude pouze ve výběrové navigaci, kde bude možno přepínat z Tratí a Kategorií,

náležejících k závodu. Rovněž zde bude nad tabulku s výsledky doplněno tlačítko pro export výsledkové tabulky do formátu xls a tlačítko pro zobrazení všech mezičasů u závodníků. Proklikem na jméno závodníka ve výsledkové tabulce bude možné zobrazit konkrétní detaily závodníka (viz obr. 4.7). U detailů závodníka budou uvedeny veškeré dostupné údaje v přehledné formě.

Obr. 4.7: Návrh vzledu Detailu závodníka



Zdroj: Vlastní zpracování v programu Balsamiq Mockups for Desktop

4.5.1 Layout Page

Protože se jednotlivé stránky svým základním vzhledem a rozvržením neliší, bylo potřeba zabezpečit toto uniformní rozložení na všech stránkách. K tomuto účelu se využila vzorová šablona (Layout page), kde se definuje fixní základní vzhled stránky. Tato rovněž obsahuje výrazové elementy, například `@RenderBody` pro generování konkrétního obsahu daných stránek. V šabloně jsou rovněž zahrnuty odkazy na veškeré skripty, potřebné pro chod aplikace.

Strukturování stránky je definováno pomocí nových HTML 5 atributů popsaných v části 2.6.1.

V šabloně je rovněž definován odkaz na font Open Sans, který stránky využívají. Pro jeho začlenění byly využity služby Google CDN.

```
<link href=
'http://fonts.googleapis.com/css?family=Open+Sans:400,600,700&subset=latin,
latin-ext' rel='stylesheet' type='text/css'>
```

4.6 Telerik UI

Jak již bylo dříve uvedeno, pro funkční složky webové aplikace je využito Telerik komponent uživatelského rozhraní. Příklad použití pro datovou tabulku kalendáře akcí ve View *KalendarAkci.cshtml* vypadá následovně:

```
@(Html.Kendo().Grid<_123Online.WebUI.Models.AkceViewModel>()
      .Name("KalendarAkci")
      .Columns(columns =>
      {
         columns.Bound(r => r.TypSportuId).Title("Sport").Filterable(false)
            .ClientTemplate(…);
         …
      })
      .DataSource(dataSource => dataSource
            .Ajax()
            .Read(read => read.Action("Akce_Read", "KalendarAkci")
            …
        )
       .Sortable()
       .Pageable(pg => pg.Messages(msg => msg
           .Page("Stránka")
           …)
           )
       .Filterable(filterable => filterable
                                    .Operators(operators => operators
                                        .ForString(str => str.Clear()
                                         .StartsWith("Začíná na")
                                         …)
                 )
)
```

Pomocí parametru `.Columns` se provádí vázání jednotlivých sloupců na data. Jeho vlastnost `.ClientTemplate` umožňuje vytvořit šablonu pro zobrazování sloupce. Pomocí této šablony jsou vytvářeny odkazy na webové stránky závodu ve formě obrázku, odkazy na detail akce atd. Následující ukázka kódu uvádí příklad šablony provádějící přiřazování obrázků typu sportu podle čísla, které má přiřazené v číselníku:

```
.ClientTemplate("<img src='" + Url.Content("~/Images/Ikony/") +
                "#:data.TypSportuId#.png' alt='#: data.TypSportuId #' />")
```

Parametr `.DataSource` určuje datový zdroj komponenty. Jak je z kódu patrné, je zde využit Ajax, který zabezpečuje, že při řazení, filtrování a stránkování se nebude znovunačítat celá stránka, ale pouze změna dat v tabulce. Pro zabezpečení funkčnosti těchto funkcí není zapotřebí vytvářet nové metody. Telerik Gridy mají tuto funkcionalitu již zabudovanou a stačí ji pouze aktivovat pomocí parametrů `.Sortable()` pro globální povolení řazení, `.Filterable()` pro globální povolení filtrování a `.Pageable()` pro umožnění stránkování. Filtrování a řazení lze lokálně zakázat u dat, u kterých buď nechceme, aby bylo uživateli umožňeno jejich provádění, nebo nemá smysl tyto tytu funkce povolovat z důvodu povahy dat. Konkrétně je filtrování například zakázáno u sloupce sport, neboť tato možnost je již zahrnuta ve výběrové navigaci.

Lokalizaci nabídky filtrování a stránkování je nutné provést zapsáním českých alternativ do kódu v parametrech jednotlivých komponent. U filtrování jde o vlastnost `.Operators()`, u stránkování `.Messages()`.

Pro českou lokalizaci komponent (kalendáře u filtrování dat, číselný formát, …) je dále potřeba nastavit správný skript určující kulturu, pro kterou je stránka určena:

```
<script type="text/javascript">
      kendo.culture(„cs-CZ");
</script>
```

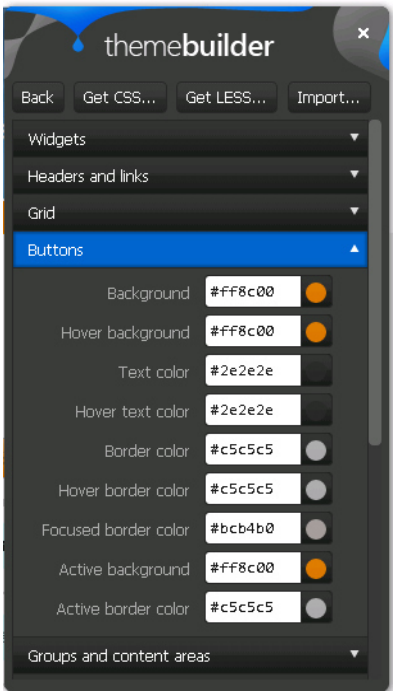
4.6.1 Vzhled Komponent

Vzhled komponent lze definovat pomocí nástroje Kendo UI Theme Builder. Nástroj je dostupný na adrese: http://demos.telerik.com/kendo-ui/themebuilder/web.html. Nespornou výhodu je možnost jeho uložení do záložek internetového prohlížeče a poté jeho přímé použití jako bookmarkletu[9] na budované aplikaci, což umožňuje pohodlnější a jednodušší stylování vzhledu. Bookmarklet umožňuje měnit pouze barevné schéma, jiné změny nejsou možné.

[9] **Bookmarklet** je kousek javascriptového kódu uložený v oblíbených položkách (záložkách). Dovoluje uživateli provést na stránce určitý javascript, který autor stránky nezamýšlel. Bookmarklet se odvozuje od slova bookmark = záložka (přípona let znamená, že něco provádí). Někdy se jim také říká favelety. Favelet je odvozen od slova favorite = oblíbený. [9]

Výsledný generovaný CSS kód se poté zkopíruje ze záložky *Get CSS* a připojí jako CSS styl k aplikaci, v našem případě *kendo.css* ve složce *Content*.

Obr. 4.8: Nástroj Telerik Kendo-UI Theme Builder



Zdroj: http://demos.telerik.com/kendo-ui/themebuilder/web.html

4.7 Sekvenční diagram Akce a jejího zobrazení v kalendáři a ve výsledcích

Následující sekvenční diagram nám přibližuje fungování aplikace. V případě vytvoření nové akce v interním evidenčním systému manažerem a změně příznaku zobrazení na webu 123Online na *ano* se uživateli zobrazí tato akce v *Kalendáři akcí*. Poté co proběhně v daný termín měření závodu, provede odpovědný časoměřič export konečných výsledků do databáze. Nyní se již tato akce nadále neobjevuje v *Kalendáři akcí*, ale zobrazuje se uživateli v seznamu akcí s výsledky v sekci *Archiv výsledků*, kde má proklikem na název akce v tabulce možnost zobrazit kromě detailnějších informací o akci rovněž konkrétní naměřené výsledky. Funkčnost tohoto zobrazování je založena na faktu, že s každým exportem dat je provedeno založení tratě závodu v tabulce *WwwTrack.* Rovněž pokud se provádí pouze nahrání PDF výsledkových listin (viz 4.8.2) je potřeba založení nové tratě závodu a je tak zabezpečeno zobrazení akce v Archivu výsledků.

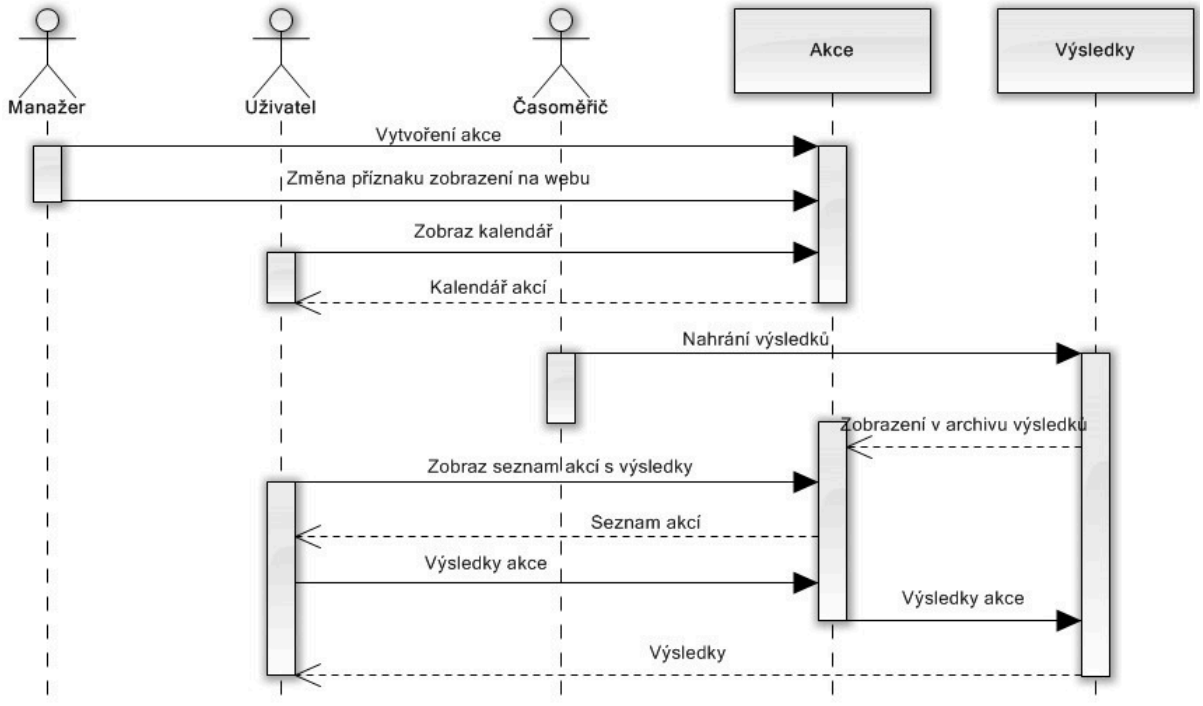
Tento algoritmus ovšem nefunguje, respektive je upraven pro sprotovní události typu atletika. Tyto akce (respektive většina akcí) nejsou měřeny pomocí softwaru Onlinesport a tudíž zde nedochází k exportu výsledků do databáze. Pro tyto akce je nastaveno zobrazení

v seznamu akcí s výsledky podle aktuálního data. V případě že aktuální datum je rovno datu konání této akce, dojde k jejímu zobrazení v sekci *Archiv výsledků* a již dále se nezobrazuje v *Kalendáři akcí.*

Následující ukázka kódu představuje výběrový příkaz akcí ve formátu lambda expression[10] pro archiv výsledků:

```
akce = akceRepository.Akce
                    .Join(akceRepository.Track,
                              a => a.Id,
                              t => t.IdEvent,
                              (a, t) => new { a, t })
                    .Where(p => p.a.Id == p.t.IdEvent)
                    .Select(p => p.a)
                    .Union(akceRepository.Akce
                        .Where(p => p.TypSportuId == 6)
                        .Where(p => p.DatumOd < timeNow));
```

Obr. 4.9 Sekvenční diagram



Zdroj: Vlastní zpracování v programu Software Ideas Modeler

[10] C# Lambda Expression: http://www.dotnetperls.com/lambda

4.8 Zobrazování výsledků

Zobrazování naměřených výsledků je stěžejní funkcionalitou webové aplikace. Poté co je provedeno změření závodu a prostřednictvím softwaru pro zpracování výsledků je proveden export dat do databáze je možno zobrazovat výsledky jednotlivých závodů. Na stránku s výsledky je možné se dostat přes proklik na název závodu na úvodní stránce aplikace v tabulce *Aktuální výsledky*, případně stejným způsobem v sekci *Archiv výsledků.*

Při prvním zobrazení, případně pokud není vybrána žádná trať dojde k načtení první trati z modelu na základě vybraného závodu:

```
if (selectedTrack == 0)
            {
                var track = repository.getTrack(selectedEvent);
                ViewBag.SelectedTrack = track;
                selectedTrack = (int)track;
            }
```

Poté na základě vybraného závodu (*selectedEvent*), vybrané trati (*selectedTrack*) a vybrané kategorii (*selectedCategory*) se vytvoří ViewModel, který je poté navrácen jako JSON objekt:

```
var results = repository.Results(selectedEvent, selectedTrack,
                                 selectedCategory);

DataSourceResult result = Results.ToDataSourceResult(request, res =>
                                                 new ResultListViewModel
                    {
                        Name = res.Name,
                       Bib = res.Bib,
                        …
                    });

return Json(result, JsonRequestBehavior.AllowGet);
```

Na stránce s výsledky závodu je poté těmito daty naplněn Telerik Kendo UI Grid:

```
@(Html.Kendo().Grid<_123Online.WebUI.Models.ResultListViewModel>()
```

Naměřené mezičasy závodníka se zobrazují pod jeho výsledným časem v závodě. Je zde vložen vnořený *Kendo().ListView<_123Online.WebUI.Models.SplitViewModel>*, který je naplněn na základě *SplitViewModel* a jednotlivé mezičasy se se závodníky párují na základě *Id* závodníků *(WwwRacer.Id = WwwRacerSplit.RacerId):*

```
dataSource.Read(read => read.Action("Splits_Read", "VysledkyAkce",
          new { racer = "#=Id#", selectedTrack = ViewBag.SelectedTrack}));
```

Protože je tento ListView vložen jako *ClientTemplate(),* musí se kromě ostatního nastavit ještě parametry *.ToClientTemplate()* a *.ToHtmlString(),* aby došlo k jeho korektní translaci a správnému zobrazení.

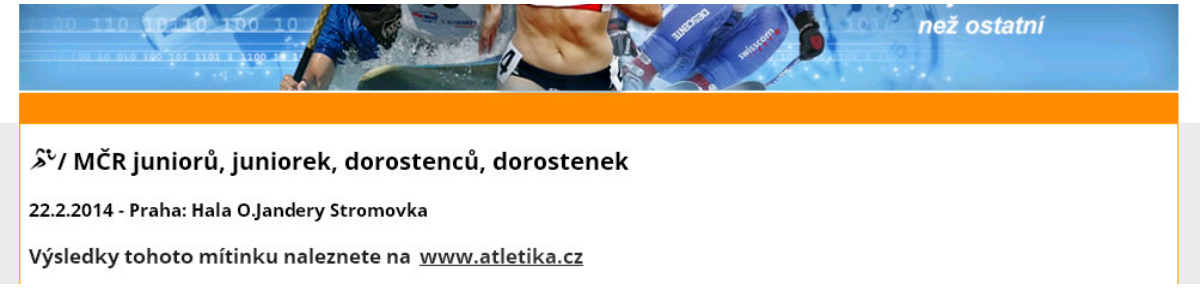
Mezičasy se sice načtou vždy, defaultně jsou ovšem skryty. Stisknutím tlačítka *Zobrazit/skrýt* mezičasy v hlavičce výsledkové tabulky se spustí jQuery kód, který přepíná u vnořeného LisView třídu *noview* a přepíná tak jejich zobrazení a skrytí*:*

```
onclick="$('.splits').toggleClass('noview');"
```

Obr. 4.10 Ukázka zobrazení výsledků s mezičasy

V případě atletických mítinků, které nejsou měřeny softwarem Onlinesport, dojde při prokliku na název akce k otevření stránky s detailnějšími informacemi o události bez datové tabulky a navigaci pro výběr tratě a kategorie. Místo nich zde bude odkaz na web www.atletika.cz, kde uživatelé naleznou požadované výsledky:

Obr. 4.11 Ukázka zobrazení výsledků atletického mítinku



4.8.1 Detail závodníka

Proklikem na jméno závodníka se uživatel dostane na stránku zobrazující detaily závodníka. Jsou zde přehledně zobrazeny veškeré podrobnosti, které se vztahují k závodníkovi a jeho účasti v závodě. Kromě jeho registračních údajů, výsledných pozic a časů startu a dokončení závodu se zde zobrazují rovněž jeho naměřené mezičasy. Pro vytvoření tohodo detailu závodníka bylo přistoupeno z důvodu možné nižší přehlednosti výsledkové tabulky v případě většího počtu mezičasů. Některé závody rovněž probíhají závoděním na kola na krátkých tratích a zobrazením časů kol ve výsledkové tabulce by se stala poněkud nepřehlednou a velmi rozsáhlou.

Rovněž v budoucnu může dojít k rozšíření aplikace a v této sekci mohou přibýt další položky, případně funkcionality, které se již nevejdou do celkové výsledkové tabulky.

4.8.2 PDF výsledky

PDF výsledky a další dokumenty spojené se závodem se ručně nahrávají na FTP server a do databáze (tabulka *WwwDocuments*) jsou ukládány odkazy na tyto soubory, názvy souborů a klíče *IdTrack* pro spárování dokumentů s konkrétní tratí. Dokumenty se tedy nepárují ke konkrétní akci, ale ke konkrétní trati závodu.

Pro zobrazení PDF výsledků u závodů je vytvořen nový ViewModel - *DocumentsListViewModel* pro dokumenty. Na stránce s výsledky je poté zavoláno vykonání *PartialView* silně typovaného na tento model a pomocí cyklu *foreach* jsou vypsány veškeré dokumenty spojené s vybranou tratí z nabídky *Trať*.

```
@model _123Online.WebUI.Models.DocumentsListViewModel

@foreach (var a in Model.Documents) {
    if (a.DocumentLink.EndsWith("pdf"))
    {
        <a  class="document-pdf" href="@(a.DocumentLink.ToString())"
                        target="_blank">@a.DocumentName.ToString()</a>
    }
…
```

Kromě PDF výsledků je takto možné uživatelům nabídnout ke stažení i další typy dokumentů, například výsledkové listiny ve formátu XLS nebo XLSX (viz obr. 4.10).

4.9 Export Výsledků do excelu

Export výsledkových tabulek do excelu je dalším důležitým funkčním prvkem aplikace. Pro zajištění potřebné funkcionality je nejprve nutné nainstalovat balíček NPOI[11] a jeho závislý balíček SharpZipLib. NPOI je .NET verze projektu POI Java. Jde o open source projekt, který pomáhá při čtení nebo zápisu XLS, DOC a PPT souborů. Umožňuje generování sestavy aplikace Excel bez sady Microsoft Office nainstalované na serveru. Instalace se provede buď přes administrační rozhraní balíčků NuGet[12], případně zadáním příkazu `Install-Package NPOI` do konzole správce balíčků.

Aplikace provádí export dat z výsledkové tabulky na základě výběru v navigaci. Exportuje se tedy konkrétní data z gridu. Je možné exportovat výsledky z dané tratě absolutně, případně pouze výsledky vybrané kategorie.

Hlavička sešitu je fixována, aby bylo zamezeno jejímu skrolování. Rovněž je zde nastavena možnost filtrování výsledků.

```
sheet.CreateFreezePane(0, 1, 0, 1);
var myTemplateRange = new CellRangeAddress(0, 220, 0, 7);
sheet.SetAutoFilter(myTemplateRange);
```

Poté se provede naplnění listu s hodnotami dat z výběru:

```
foreach (Domain.Entities.Akce akce in dataAkce)
      {
      var row = sheet.CreateRow(rowNumber++);
      row.CreateCell(0).SetCellValue(akce.DatumOd.ToShortDateString());
      …
      }
```

Výsledný exportovaný soubor bude mít název podle vybraného závodu a vybrané tratě:

```
string filename = ViewBag.EventName +"_"+ model.TrackName + ".xls";
```

[11] NPOI: https://www.nuget.org/packages/NPOI/

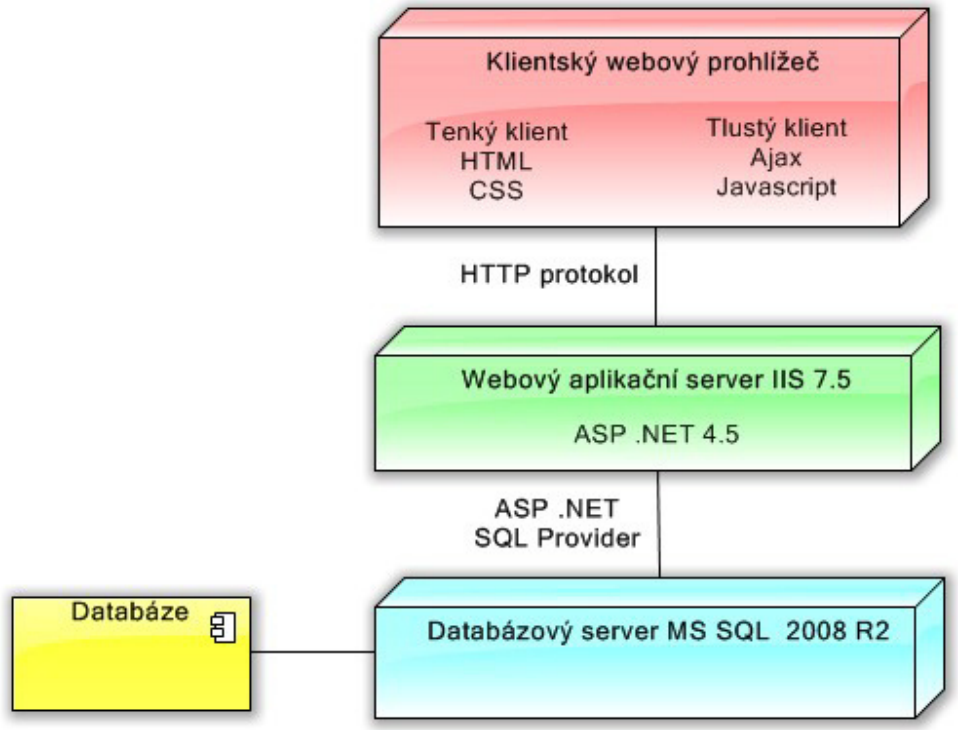
[12] **NuGet** je správce balíčků pro vývojové platformy Microsoft včetně .NET. Jde o centrální úložiště balíčků používané všemi autory balíčků: http://www.nuget.org/

4.10 Diagram Nasazení

Aplikace je nasazena na serveru, na němž běží webový aplikační server IIS 7.5. Nad ním pracuje technologie ASP .NET 4.5, která zajišťuje zpracování serverových prvků implementovaných do ASP .NET stránek. Rovněž vykonává události vyvolané klientem prostřednictvím http protokolu.

Pro práci s daty je na serveru onlinesystem.dbaserver.net implementován databázový server MS SQL 2008 R2. Komunikace s aplikačním serverem je zajištěna prostřednictvím služby ASP .NET SQL Provider.

Obr. 4.12: Diagram nasazení



Zdroj: Vlastní zpracování v programu Software Ideas Modeler

4.11 Zhodnocení realizovaného systému

Aplikace se v současné době (duben 2014) připravuje na své finální spuštění do ostrého provozu. Na většinu běžně měřených základních závodů je funkcionalita již dostačující a umožňuje zobrazovat potřebné výsledkové tabulky. Dosud však aplikace obsahuje pouze výsledky z několika závodů, neboť export dat ze softwaru pro zpracování výsledků ještě není kompletně implementován.

Aplikace však dosud není zcela dokončena. Je stále rozšiřována o nové funkce zobrazování různých druhů závodů a jejich výsledků. Chybí zde zakomponovat funkcionalita pro týmové výsledky, která bude vyžadovat další rozšíření databázové struktury.

V původním zadání nebyla požadována administrace systému. Veškeré administrační zležitosti akcí měly být řešeny prostřednictvím interního administračního systému. V průběhu tvorby se dospělo k závěru, že ne všechny záležitosti lze takto vyřešit. Jednoduché administrační rozhraní by bylo vhodno doplnit pro dokumenty k závodům. V současné době se provádí ruční nahrávání těchto dokumentů na FTP server a odkazy na tyto dokumenty se ručně vkládají do databáze.

5 Závěr

Předmětem této diplomové práce bylo vytvoření, realizace a implementace webové aplikace výsledkového servisu sportovních událostí pro společnost On line System. Společnost se zabývá poskytováním služeb v oblasti časomíry a postrádala svůj vlastní systém pro prezentaci naměřených výsledků. K tvorbě aplikace byl využit framework ASP .NET MVC 4 společnosti Microsoft založený na návrhovém vzoru Model-View-Controller s využitím komponent uživatelského rozhraní společnosti Telerik. Tyto komponenty jsou velmi sofistikované a umožňují rychle a jednoduše vytvořit potřebné uživatelské rozhraní bez potřeby psaní obsáhlého kódu.

Podařilo se vytvořit webovou aplikaci, napojenou na interní administrační systém pro správu akcí. Aplikace zobrazuje kalendář akcí, které ještě neproběhly a nebylo provedeno jejich měření. Rovněž je zde sekce archivu výsledků kde uživatelé naleznou všechny závody u kterých jsou dostupné výsledky. Výsledky závodu se zobrazují v přehledné tabulce a také je uživatelům umožněno zobrazit nebo stáhnout oficiální PDF výsledkové listiny, případně další dokumenty spojené s konkrétním vybraným závodem. Uživatelům je rovněž umožněno provést export vybraných výsledků z tabulky do excelu. Požadované funkční požadavky a tedy i cíle práce na aplikaci lze označit za dosažené, a to i přesto, že aplikace dosud není zcela dokončená a pokračuje se na jejím vývoji.

Diplomová práce je rozdělená do několika částí. Úvodní část, Metodická východiska a nástroje, seznamuje se základními pojmy a technologiemi, které jsou v kontextu s předmětem diplomové práce. Je zde popsána technologie ASP .NET, aplikační rámec MVC 4 a další technologie s ním spolupracující a tvořící nedílnou část aplikace, jako jsou HTML5, Entity Framework, AJAX a další.

Druhá část zmiňuje současný stav výsledkového servisu a způsoby prezentace naměřených výsledků závodníkům. Dále jsou zde definovány požadavky na nový systém a analýza budoucího stavu.

Poslední část čtenáře seznamuje se samotnou tvorbou a realizací webové aplikace. Nastiňuje základní mechanismy fungování, definici vzhledu a uživatelského prostředí a v závěru zhodnocuje vytvořenou aplikaci a nastiňuje možné budoucí rozšíření, které by

mohlo zvýšit funkcionalitu a použitelnost i pro další typy závodů a různé možnosti zobrazování potřebných výsledků.

Výstupem diplomové práce je webová aplikace, která spolupracuje s databázi firemního systému sloužícímu manažerovi pro interní administraci jednotlivých akcí a umožňuje zobrazování výsledků měřených závodů. Výsledky jsou exportovány ze systému pro zpracování měřených dat Onlinesport.

Seznam použité literatury

Tištěné monografie

[1] FOWLER, Martin. *Destilované UML*. Praha: Grada, 2009. 176 s. ISBN 978-80-247-2062-3.

[2] FREEMAN, Adam. *Pro ASP .NET MVC 4*. 4th. Ed. Berkley: Apress, 2012. 756 s. ISBN 978-1-4302-4236-9.

[3] GALLOWAY, Jon et al. *Professional ASP.NET MVC 4*. Indianapolis: Wiley, 2012. 504 s. ISBN 978-1-118-42432-2.

[4] LACKO, Luboslav. *PHP a MySQL: Hotová řešení.* 1. Vyd. Brno: CP Books, 2005. 304s. ISBN: 80-251-0397-8.

[5] MacDONALD, M. a SZPUSZTA, M. *Pro ASP.NET 4.0 in C# 2010*. 4th. Ed. Berkeley : Apress, 2010. Str. 1616. ISBN 978-1-4302-2529-4.

[6] SKLENÁK, Vilém a kol. *Data, informace, znalosti, a Internet* Praha: C. H. Beck 2001. 507 s. ISBN 80-7179-409-0.

[7] ŠKULTÉTY, Rastislav. *JavaScript: Programujeme internetové aplikace.* 2. Akt. Vyd. Brno: Computer press, 2004. 224 s. ISBN: 80-251-0144-4.

Elektronické zdroje

[8] HUNT, Lachlan. *Seznámení s HTML 5* [online]. [19. 3. 2014]. Dostupné z: http://interval.cz/clanky/seznameni-s-html-5/

[9] JANOVSKÝ, Dušan. *Bookmarklet.* [online]. [7. 4. 2014]. Dostupné z: http://www.jakpsatweb.cz/javascript/bookmarklet.html

[10] LOKUGE, Sampath. *Top 10 new features in ASP NET MVC 4* [online]. [2. 4. 2014]. Dostupné z: http://sampathloku.blogspot.cz/2013/05/top-10-new-features-in-aspnet-mvc-4.html

[11] MANAGEMENTMANIA. *Webová aplikace* [online]. [19. 2. 2014]. Dostupné z: https://managementmania.com/cs/webova-aplikace-web-application

[12] MICROSOFT. *.edmx File Overview (Entity Framework)* [online]. [24. 3. 2014]. Dostupné z: http://msdn.microsoft.com/en-us/library/vstudio/cc982042(v=vs.100).aspx

[13] MICROSOFT. *(LINQ – Language Integrated Query)* [online]. [19. 3. 2014]. Dostupné z: http://msdn.microsoft.com/cs-cz/library/bb397926.aspx

[14] PURCHART, Václav. *Dependency Injection: předávání závislostí* [online]. [24. 3. 2014]. Dostupné z: http://www.zdrojak.cz/clanky/dependency-injection-predavani-zavislosti/

[15] ŠIMEČEK, Martin. *HTML5 – nové vlastnosti* [online]. [19. 3. 2014]. Dostupné z: http://programujte.com/clanek/2010082200-html5-nove-vlastnosti/

[16] TELERIK. UI for ASP.NET MVC: *Key Features* [online]. [23. 3. 2014]. Dostupné z: http://www.telerik.com/aspnet-mvc

[17] W3C. *Differences from HTML 4* [online]. [19. 3. 2014]. Dostupné z: https://rawgit.com/whatwg/html-differences/master/Overview.html

[18] W3SCHOOLS. *CSS3 Introduction.* [online]. [19. 3. 2014]. Dostupné z: http://www.w3schools.com/css/css3_intro.asp

Seznam zkratek

AJAX	Asynchronous JavaScript and XML
API	Application Programming Interface
ASP .NET	Active Server Pages .NET
CDN	Content Delivery Network
CLR	Common Language Runtime
CSS	Cascading Style Sheets
C#, J#	Programovací jazyky
DI	Dependency Injector
DTD	Document Type Definition
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
IIS	Internet Information Services
IL	Intermediate Language
IOC	Inversion of Control
JavaScript	Skriptovací jazyk
JIT	Just in Time
JSON	JavaScript Object Notation
LINQ	Language Integrated Query
MS SQL	Microsoft Structured Query Language
MVC	Model View Controller
NPOI	.NET Poor Obfuscation Implementation
OOP	Object Oriented Programming
PDF	Portable Document Format
PHP	Hypertext Preprocessor
POI	Poor Obfuscation Implementation
RFID	Radio Frequency Identification
UHF	Ultra High Frequency
UI	User Interface
UML	Unified Modeling Language
WWW	World Wide Web

XHTML	eXtensible HyperText Markup Language
XLS	přípona souborů specifikace Office Open XML vytvořených v aplikaci Microsoft Excel
XML	eXtended Markup Language

Prohlášení o využití výsledků diplomové práce

Prohlašuji, že

- jsem byl seznámen s tím, že na mou diplomovou práci se plně vztahuje zákon č. 121/2000 Sb. – autorský zákon, zejména § 35 – užití díla v rámci občanských a náboženských obřadů, v rámci školních představení a užití díla školního a § 60 - školní dílo;
- beru na vědomí, že Vysoká škola báňská – Technická univerzita Ostrava (dále jen VŠB-TUO) má právo nevýdělečně, ke své vnitřní potřebě, diplomovou práci užít (§ 35 odst. 3);
- souhlasím s tím, že diplomová práce bude v elektronické podobě archivována v Ústřední knihovně VŠB-TUO a jeden výtisk bude uložen u vedoucího diplomové práce. Souhlasím s tím, že bibliografické údaje o diplomové práci budou zveřejněny v informačním systému VŠB-TUO;
- bylo sjednáno, že s VŠB-TUO, v případě zájmu z její strany, uzavřu licenční smlouvu s oprávněním užít dílo v rozsahu § 12 odst. 4 autorského zákona;
- bylo sjednáno, že užít své dílo, diplomovou práci, nebo poskytnout licenci k jejímu využití mohu jen se souhlasem VŠB-TUO, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly VŠB-TUO na vytvoření díla vynaloženy (až do jejich skutečné výše).

V Ostravě dne 25.4.2014 ..... Fluksa Jiří

Bc. Jiří Fluksa

Seznam příloh

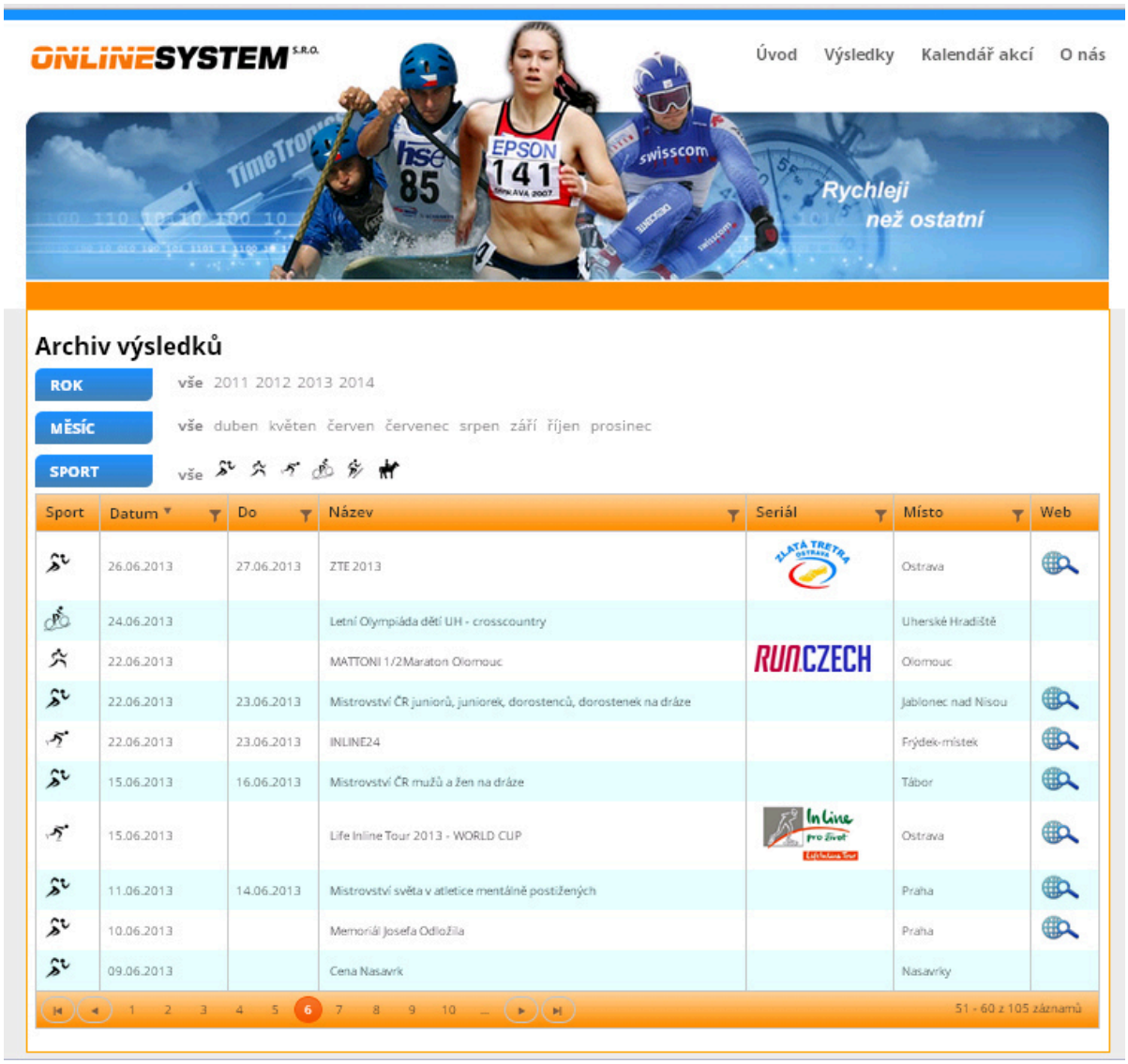
Příloha A

Obsah CD

- Text diplomové práce ve formátu pdf,
- zdrojové soubory aplikace,
- ukázka exportovaných výsledků ve formátu xls,
- soubor readme.txt s informacemi k aplikaci.

Příloha B

Ukázky webové aplikace



ONLINESYSTEM s.r.o.

Úvod Výsledky Kalendář akcí O nás

/ XXX.Běh Golčovým Jeníkovem

20.4.2014 - Golčův Jeníkov: před radnicí

Trať XXX.Běh Městem Jarmily Kratochvílové

Kategorie vše | muži | ženy

A - Muži do 39 let | B - Muži 40-49 let | C - Muži 50-59 let | D - Muži nad 60 let | E - Ženy do 34 let | F - Ženy nad 35 let

Výsledky Absolutní poř. | Absolutní poř. | Kategorie

Zobrazit/skrýt mezičasy

Exportovat výběr do Excelu

#	v kat.	v pohl.	číslo	jméno	klub	stát	čas	čas čipový
1	1	1	14	Joel Moina MWHNGI	Benedek Team		00:44:57	
2	2	2	26	Haron Kiptoo Sirma	UNIPES TEAM		00:45:01	
3	3	3	27	Moses Kipkosgei Bowen	UNIPES TEAM		00:45:43	
4	4	4	16	Elisha Kiprotich Sawe	Benedek Team		00:47:05	
4	4	4	5	Olexander Babarika	JM demolex Bardějov		00:47:05	
6	6	6	4	Dmytro Laschyn	JM demolex Bardějov		00:48:31	
7	7	7	15	Wilson Wambugugr Kamau	Benedek Team		00:49:38	
8	8	8	20	Timashov Volodymyr	achiles leszno		00:49:51	
9	9	9	19	Pototskyi Anton	achiles leszno		00:50:01	
10	10	10	17	Tamas Nagy	Benedek Team		00:50:11	